

AD-A174 137

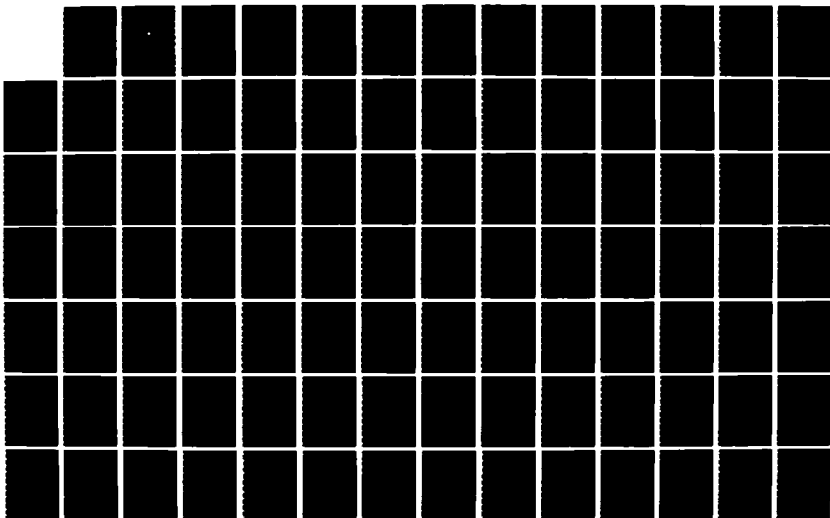
IMPLEMENTATION OF A PERSONNEL DATABASE SYSTEM
PERFORMING THE ANNUAL REASS (U) NAVAL POSTGRADUATE
SCHOOL MONTEREY CA I G STROUZAS 20 JUN 86

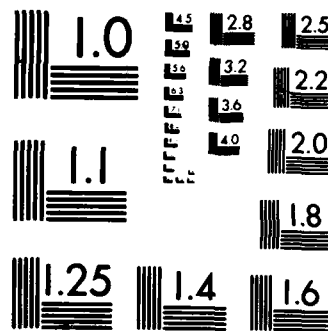
1/3

UNCLASSIFIED

F/G 5/9

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

2

AD-A174 137

NAVAL POSTGRADUATE SCHOOL

Monterey, California



DTIC
ELECTE
NOV 20 1986
B

THESIS

IMPLEMENTATION OF A PERSONNEL DATABASE
SYSTEM PERFORMING THE ANNUAL REASSIGNMENT
OF THE OFFICERS OF A BRANCH DIRECTORATE OF
THE HELLENIC ARMY GENERAL STAFF

by

Ioannis G. Strouzas

June 1986

Thesis Advisor:

L. C. Rawlinson

Approved for public release; distribution is unlimited.

DTIC FILE COPY

86 11 10 000

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION			1b. RESTRICTIVE MARKINGS UNCLASSIFIED		
2a SECURITY CLASSIFICATION AUTHORITY			3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.		
2b DECLASSIFICATION/DOWNGRADING SCHEDULE					
4 PERFORMING ORGANIZATION REPORT NUMBER(S)			5. MONITORING ORGANIZATION REPORT NUMBER(S)		
6a. NAME OF PERFORMING ORGANIZATION Naval Postgraduate School		6b. OFFICE SYMBOL (If applicable) 52		7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School	
6c. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000			7b. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000		
8a NAME OF FUNDING/SPONSORING ORGANIZATION		8b. OFFICE SYMBOL (If applicable)		9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER	
8c. ADDRESS (City, State, and ZIP Code)			10 SOURCE OF FUNDING NUMBERS		
			PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.
			WORK UNIT ACCESSION NO.		
11 TITLE (Include Security Classification) UNCLASSIFIED - Implementation of a Personnel Database System Performing the Annual Reassignment of the Officers of a Branch Directorate of the Hellenic Army General Staff					
12 PERSONAL AUTHOR(S) Ioannis G. Strouzas					
13a TYPE OF REPORT Master's Thesis		13b TIME COVERED FROM _____ TO _____		14 DATE OF REPORT (Year, Month, Day) 1986 June 20	
15 PAGE COUNT 203					
16 SUPPLEMENTARY NOTATION					
17 COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)		
FIELD	GROUP	SUB-GROUP	Database System, Annual Reassignment Officer, Branch Directorate		
19 ABSTRACT (Continue on reverse if necessary and identify by block number) The Branch Directorates of the Hellenic Army General Staff (HAGS) currently perform the annual reassignment of their officers manually. The author proposes an automated system to perform this function, as well as other functions concerning personnel management, using dBASE III with a micro-computer which is fully software-compatible with the IBM/PC or IBM PC/XT. Source programs and sample reports are included.					
20 DISTRIBUTION/AVAILABILITY OF ABSTRACT <input type="checkbox"/> UNCLASSIFIED/UNLIMITED <input type="checkbox"/> SAME AS RPT <input type="checkbox"/> DTIC USERS			21 ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED		
22a NAME OF RESPONSIBLE INDIVIDUAL CDR L. C. Rawlinson			22b TELEPHONE (Include Area Code) 408 646-2735		22c OFFICE SYMBOL 52Rv

Approved for public release, distribution unlimited

Implementation of a Personnel Database System
Performing the Annual Reassignment of the Officers of a
Branch Directorate of the Hellenic Army General Staff

by

Ioannis G. Strouzas
Major, Hellenic Army
B.A., Hellenic Army Academy, 1969

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
June 1986

Author:



Ioannis G. Strouzas

Approved by:


L. C. Rawlinson, Thesis Advisor


G. S. Baker, Second Reader


Vincent Y. Lum, Chairman,
Department of Computer Science


Kneale T. Marshall,
Dean of Information and Policy Sciences

ABSTRACT

The Branch Directorates of the Hellenic Army General Staff (HAGS) currently perform the annual reassignment of their officers manually. The author proposes an automated system to perform this function, as well as other functions concerning personnel management, using dBASE III with a micro-computer which is fully software-compatible with the IBM/PC or IBM PC/XT. Source programs and sample reports are included.

[illegible]

TABLE OF CONTENTS

I.	INTRODUCTION	10
A.	PREFACE	10
B.	COMPUTERS IN THE HELLENIC ARMY	11
C.	DATABASE SYSTEMS VS MANUAL SYSTEMS	11
D.	GENERAL OVERVIEW OF A DATABASE PROCESSING SYSTEM	12
	1. Definition and Basic Terminology	12
	2. Architecture of a Database System	15
	3. Database Systems vs Traditional File Systems	15
	4. Data Models	19
E.	dBASE III CONCEPTS	23
	1. Features of dBASE III	23
	2. Limitations of dBASE III	24
II.	ANALYSIS	25
A.	PROBLEM DEFINITION	25
	1. Organization Overview	26
	2. Description of the Present Situation	29
B.	JUSTIFICATION OF A COMPUTERIZED SOLUTION	34

C.	SYSTEM GOALS AND REQUIREMENTS	35
1.	Goals	35
2.	Requirements	35
D.	INPUT/OUTPUT INFORMATION	36
1.	Input Information	37
2.	Output Information	37
III.	DESIGN	39
A.	LOGICAL (CONCEPTUAL) DESIGN	39
1.	System Functions	39
2.	Input Design	41
3.	File Design	44
4.	Conceptual Database Structure	51
5.	Output Design	53
B.	PHYSICAL DESIGN	57
1.	File Definition/Creation	58
2.	Program Creation	58
IV.	SYSTEM IMPLEMENTATION	60
A.	HARDWARE REQUIREMENTS	60
B.	TRAINING PERSONNEL	61
C.	CONVERSION PROCEDURES	62
D.	SOFTWARE DESCRIPTION/DOCUMENTATION	62
1.	Main Program	63

2. Main-Menu and Sub-Menu Programs	64
3. Programs Supporting the Update Operations	65
4. Programs Performing the Assignment Processing	70
5. Programs Producing the Required Lists and Reports	78
6. Miscellaneous Programs	82
V. CONCLUSIONS AND RECOMMENDATIONS	85
APPENDIX A	87
A. MAIN PROGRAM	87
B. MAIN-MENU AND SUB-MENU PROGRAMS	89
C. PROGRAMS SUPPORTING THE UPDATE OPERATIONS	93
D. PROGRAMS PERFORMING THE ASSIGNMENT PROCESSING	113
E. PROGRAMS PRODUCING THE REQUIRED LISTS AND REPORTS	169
F. MISCELLANEOUS PROGRAMS	186
G. SAMPLE LISTS AND REPORTS	191
LIST OF REFERENCES	200
INITIAL DISTRIBUTION LIST	201

LIST OF FIGURES

1.	Levels of Abstraction in a Database System	16
2.	File Processing Systems	17
3.	Database Processing System	17
4.	Network Data Model	20
5.	Hierarchical Data Model	21
6.	Relational Data Model	22
7.	Organization Chart of the HAGS	26
8.	Organization Chart Of the Artillery Branch	27
9.	Functional Blocks of the System	41
10.	Main Menu	42
11.	Update Menu	42
12.	Assignment Processing Menu	43
13.	Report Generator Menu	44
14.	Structure for File OFFICER	45
15.	Structure for File SERVES	45
16.	Structure for File REQUESTS	46
17.	Structure for File ASSIGNED	46
18.	Structure for File UNITORG	47
19.	Structure for File UNIT	48
20.	Structure for File HISTORIC	48
21.	Structure for File SCHOOLS	49
22.	Structure for File SELECTED	50
23.	Structure for File USERLOG	50
24.	Structure for File USERID	51
25.	Entity/Relationship Diagram	52
26.	The Relational Database Scheme	52
27.	List of Scheduled Assignments	53
28.	List of Artillery Officers in Some Order	54
29.	List of Officers of a Specific Unit	54

30.	List of Officers of Any Desired Rank	55
31.	List of Battalion Commanders	55
32.	List of Officers Serving Outside the Branch	56
33.	Officer's Service Time Report	56
34.	Officer's Status Report	57
35.	Program Structure Chart	59

ACKNOWLEDGEMENT

A number of people assisted me during the writing of this thesis.

I would like to express my deep gratitude to CDR L. C. Rawlinson of the Department of Computer Science and CDR G. S. Baker of the Department of Computer Science, too. In addition to giving encouragement, they read through the thesis, suggested corrections and improvements.

I also want to acknowledge the understanding and support of my wife Niki, and my children Anastasia and George.

Finally, I would like to thank my sister-in-law Jan Strouzas for encouraging and helping me in various ways for turning my dream of obtaining a MSC Degree into a reality.

I. INTRODUCTION

A. PREFACE

Look around and you will see evidence that computers have become part of our daily lives. Today, computers are common place in industry, government, science, politics, and even in our homes. As more and more organizations use computers for their needs, it is necessary to use modern, systematic, and cost effective approaches for software solutions to their problems. In recent years these software solutions have clearly been motivated by the database systems approach, which is widely used in most organizations.

Database technology today, plays a central role in the computer world for the facilities and data handling capabilities they provide.

During the last decade the cost of labor has been increasing steadily with direct consequence being the parallel increase of the software costs. Meanwhile the cost of computer hardware has decreased dramatically. As David Kroenke puts it, 'The computer industry has developed the equivalent of a \$2.50 Rolls Royce that gets 2,000,000 miles per gallon [Ref. 1: p. 11].

Thus, simply stated, software has become more expensive as computer hardware has become cheaper. In 1960, the ratio of hardware over software expenditures was approximately 80 percent hardware cost to 20 percent software cost. By 1980, the ratio was reversed. By 1990, software costs will account for more than 90 percent of the amount spent on computing systems [Ref. 2: p. 8].

The above considerations lead us to select systems that achieve the best utilization of the software, and thus

motivated system designers to build advanced database systems in order to decrease software cost and obtain maximum benefit.

B. COMPUTERS IN THE HELLENIC ARMY

The leadership of the Hellenic Army, realizing the usefulness of computers, introduced them into the Army in the 1970's. Today, most of the tedious and error prone bureaucratic manual procedures are automated, and the computer is used efficiently in data processing and decision making. Of course there are a lot of procedures, especially in the area of personnel management, which have not been automated yet. One of these procedures which is still performed manually is the annual reassignment processing of officers in each Branch of the Hellenic Army General Staff (HAGS). The automation of this job constitutes the topic of this research.

C. DATABASE SYSTEMS VS MANUAL SYSTEMS

The top management in every organization, and in our case the Branch Directors of the HAGS, need accurate and timely information in order to make fast and better informed decisions.

Currently, all information required by the Director for scheduling and processing annual assignments of his officers is handled manually by the staff of the Directorate. This results in tedious and time-consuming operations, which are sometimes inaccurate.

Because of the complicated character of this job, and the continuous changes pertaining to personnel and the associated data, it is extremely difficult for the staff personnel to

process this job. Further, the Branch Director frequently does not have the necessary information to make rapid decisions.

The above problem could be overcome by developing and implementing an automated personnel database system.

The development and implementation of a personnel database processing system would provide the following advantages over the existing manual system:

1. Improved productivity, i.e. fewer people can do the same job. This is very important since we can reduce the staff personnel involved in the manual system, and use them for other productive tasks.
2. Speed, which is very important for a decision-oriented processing environment.
3. Reduced tedium. Large volume repetitious jobs can be processed easily.
4. Improved quality of decisions. Up-to-date data/information can be made available to decision makers.

D. GENERAL OVERVIEW OF A DATABASE PROCESSING SYSTEM

In this section some definitions and basic database terminology are provided, followed by a summary of database architecture and types of data models. A detailed discussion of the data models is beyond the scope of this thesis, but a brief overview is important as an introduction to dBASE III.

1. Definition and Basic Terminology

a. Database

A shared collection of interrelated data designed to meet the varied information needs of an organization.

b. Database Management System (DBMS)

A software system that carries out all user requests for data. User requests may be an update, a delete or a retrieval operation/function.

c. Database System

A system to record and maintain information that is significant to organization in the decision making process. It is also called Information System.

d. Data Definition Language (DDL)

A specialized language used for the description of the database (records and data-items). This description is stored in the Data Dictionary maintained by the DBMS.

e. Data Manipulation Language (DML)

A programming language used to formulate queries or to write application programs for data manipulation. It is also called Host Language or Query Language.

f. File (or Entity Set)

An organized collection of records representing entities of the same type.

g. Record

A unit of data representing a particular entity of a file. It consists of a number of interrelated data elements.

h. Field

A subdivision of a record containing a unit of information. It is the smallest unit of named data.

i. Key

An attribute (field) or a set of attributes whose value uniquely identify each entity in a file.

j. Relationships

A relationship among entity sets (files) is simply an ordered list of entity sets. Relationships are classified into the following three categories according to how many entities from one entity set can be associated with how many entities of another entity set [Ref. 3: p. 14]:

- (1) One-to-one Relationship. For an entity A in either set there is exactly one associated entity B of the other set. Eg. Suppose that in a database we have the entity sets DEPARTMENT and HEAD_OF_DEPARTMENT. The two sets form a one-to-one relationship since each department has only one head and each head can belong only to one department. [Ref. 3: p. 15]
- (2) One-to-many Relationship. For an entity A in either set there are (possibly) many associated entities of the other set. For example the entity sets ORDER and CUSTOMER form a one-to-many relationship since each order is related with a specific customer, while a customer may be related with more than one orders.
- (3) Many-to-many Relationship. There are no restrictions on the sets of pairs of entities that may appear in a relationship set. For example the entity sets PRODUCT and RAW_MATERIAL form a many-to-many relationship since a product may be built from more than one raw material and a raw material may be used to build more than one type of product.

2. Architecture of a Database System

It is obvious that between the computer, dealing with bits, and the end user sitting in front of a terminal managing information, there can be many levels of abstraction. The database architecture is divided into three different levels of abstraction: internal, conceptual, and external. In Figure 1 we can see the standard view-points regarding the three levels of a single database, which may be one of many databases using the same DBMS software. [Ref. 3: p. 6]

The internal view is the physical database, and resides permanently on secondary storage devices, such as disks and tapes. It should be emphasized that only the physical database exists. We may view the physical database itself at several levels of abstraction, ranging from that of records and files in a programming language such as COBOL, through the level of logical records, as supported by the operating system underlying the DBMS, down to the level of bits and physical addresses on storage devices.

The conceptual view (or schema) is an abstraction of the complete picture of an organization. A DBMS provides the Data Definition Language to specify the conceptual scheme.

The external view (or subschema) is an abstract model of a portion of the conceptual view. More commonly it is called user view.

3. Database Systems vs Traditional File Systems

Database technology allows an organization's data to be processed as an integrated whole. It reduces the need of creating and maintaining separate files for separate applications and permits users to access data more naturally. [Ref. 1: p. 1]

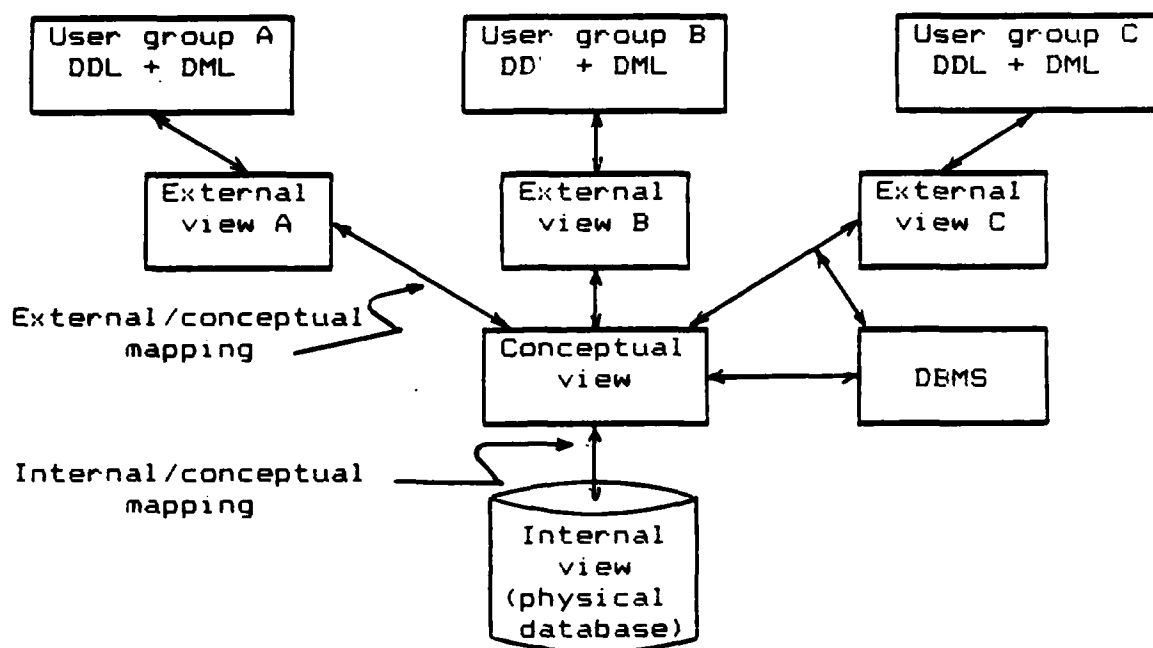


Fig. 1. Levels of Abstraction in a Database System.

To appreciate this concept, consider the systems shown in Figure 2. These are three traditional file processing systems. Each file is considered to exist independently, and each application program maintains its own files. Also there is no sharing of data among different application programs.

Figure 3 shows a database processing system. The files from the previous approach have been integrated into a database which is processed indirectly by the application programs. The new system can perform all the old functions, but the programs call upon the DBMS to access the database. The DBMS acts as a data librarian. It stores and retrieves data. Besides the data it stores in the database, the DBMS also stores a description of the format of the data. This is necessary in order for the DBMS to be able to perform its function. [Ref. 1: p. 3]

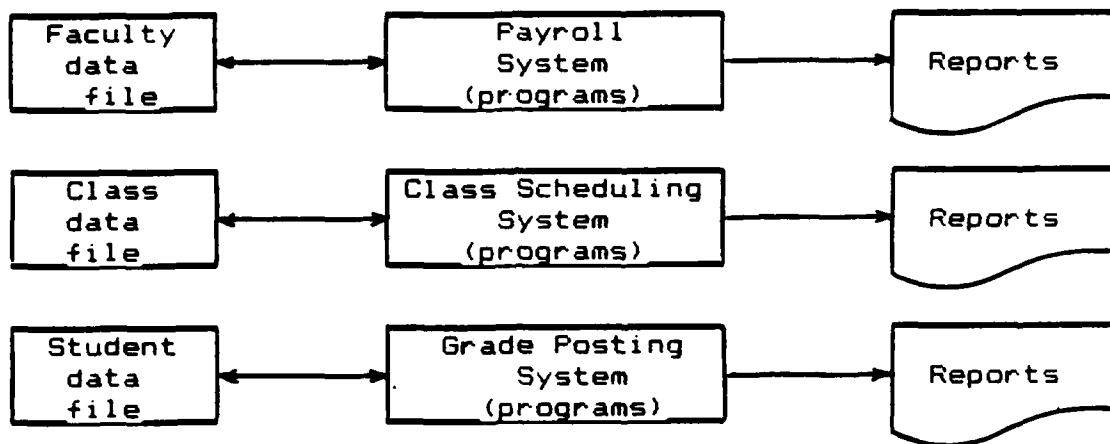


Fig. 2. File Processing Systems.

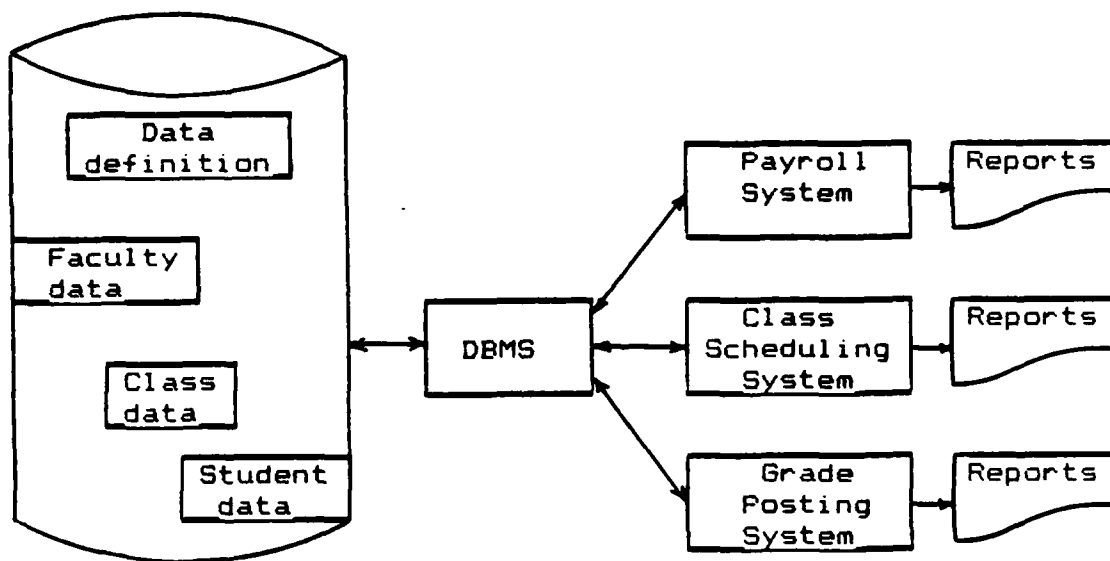


Fig. 3. Database Processing System.

The database processing approach is more beneficial than the traditional file processing approach for the following reasons:

- a. Integrates data into a single database. This feature is extremely important because it enables more information to be produced from a given amount of data. This is because DBMS allows processing of any combination of data stored in the database, and thus we can obtain more information. In the file processing system the combinations of data that can be performed are limited, since data is physically partitioned, and hence the amount of information obtained is limited. [Ref. 1: p. 3]
- b. Minimizes data redundancy. Separate and redundant data files are integrated into a single logical structure. This means that a data item is recorded once, while in the file processing system the same information can be repeated in different files.
- c. Provides data consistency. Because the data redundancy is controlled, there is less chance of inconsistencies. This is not true for a file processing system, since the data redundancy is uncontrolled.
- d. Allows sharing of data. Data can be shared by many application programs via DBMS. In the file processing approach, since every application has its own private files, there is little opportunity to share data from other application programs' files.
- e. Allows enforcement of standards. Since data is repeated only once, maintaining a standard is a lot easier.
- f. Facilitates the development of new applications. There is no need for designing, building, and maintaining anew separate files for new applications, while in the file processing approach, usually a new application must be build from scratch.
- g. Provides uniform security, privacy, and integrity controls. Some of those controls are provided directly by the DBMS (concurrency control), and others are specified by the user during the database definition (integrity rules) and the program development (security constraints). This is an immediate benefit from the sharing and integration of data.
- h. Creation of program/data independence. DBMS isolates any changes in file formats, record structure, etc. from application programs. Therefore only the DBMS and those programs that use the changed data element need to be

modified. In the file processing systems, programs interface directly with files and hence the structure of files is distributed across the programs. This distribution creates problems when a file is changed. [Ref. 1: p. 4]

- i. Facilitates data accessibility and responsiveness. DBMS provides an interactive interface to database by query language.
- j. Reduces program maintenance. This is a direct consequence from the program/data independence feature. This is not true for the file processing approach where any change in a datafile will necessitate a (possibly major) change in programs.

4. Data Models

A model is a representation of real world objects, events, and their associations in a mathematical form.

A data model is an abstract representation of the data about entities, events, activities, and their associations. The purpose of a data model is to represent data in an understandable way. The three most important data models in use today are the network, hierarchical and relational. These models are also used to categorize DBMS products. [Ref. 3: p. 18]

a. Network Data Model (NDM)

A network data model represents data as a set of record types and pairwise relationships between record types (Figure 4). Relationships that involve more than two record types are not directly permitted.

The basic data structure used in a network database is the graph. The links in the graph are bidirectional, allowing us to travel either from many to one or from one to

many. The process of following the graph links, or more generally, relationships is called navigation [Ref. 3: p. 30]. Navigation allows us to search the database and perform the basic operations (retrieve, insert, modify, or delete).

The DBMS in a network database processing system supports the use of multiple one-to-many or many-to-one relationships between the same pair of record types, but cannot support directly the use of many-to-many relationships.

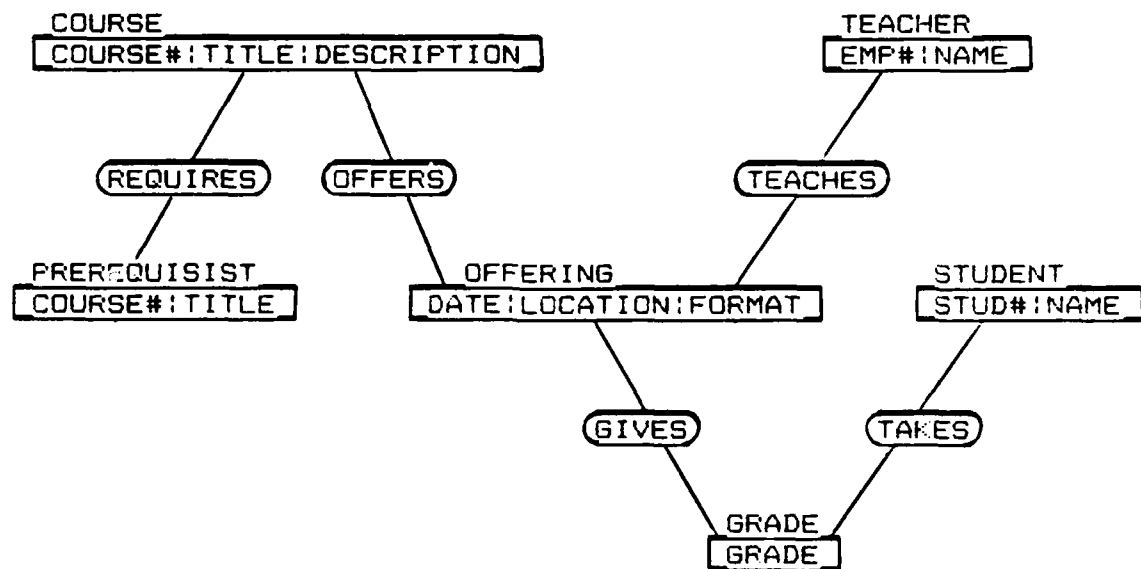


Fig. 4. Network Data Model

b. Hierarchical Data Model (HDM)

In the Hierarchical Data Model organizations are viewed as a hierarchy of positions. A hierarchical database consists of one or more trees and each tree consists of a hierarchy of records (Figure 5). Hierarchical data models are

considered as a special case of the network data model since the tree structure is a special case of the graph.

The basic operation on a hierarchical database is a tree walk, that is, given a node of the database instance, we can scan all of the descendants of a given logical record type. This allows us to insert new records, retrieve, modify, or delete existing records.

The above operation is unidirectional, that is, the links in the tree proceed from parent to child only [Ref. 3: p. 32]. For this reason HDM is considered inefficient in supporting many-to-one relationships.

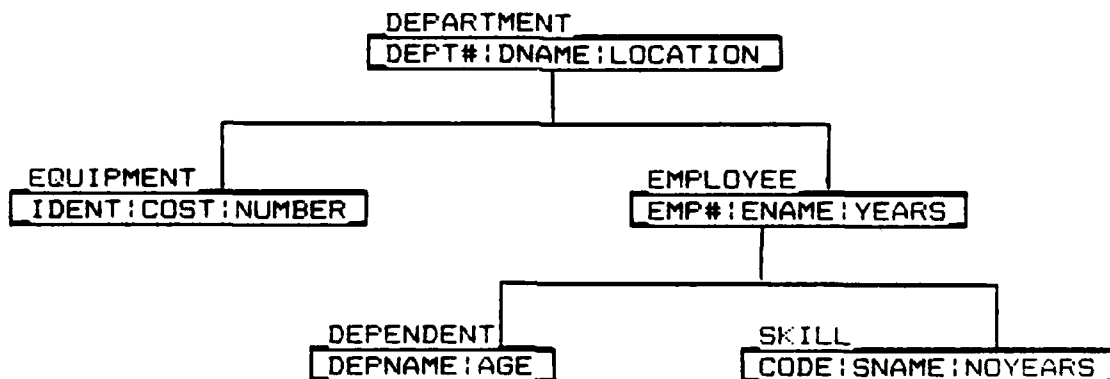


Fig. 5. Hierarchical Data Model.

c. Relational Data Model (RDM)

A relational data model differs from NDM and HDM in architecture. The data are represented as a collection of relations. Intuitively, a relation is a two dimensional table (Figure 6) representing a file. The rows of the relation are

the file records. Rows sometimes are called tuples of the relation. Each column contains values about the same attribute (field), and has a distinct name, i.e., the name of the attribute. The sequence of the rows is immaterial [Ref. 1: p. 196].

The set of attribute names for a relation is called the relation scheme. For the example in Fig. 6, the relation scheme for the relation PRODUCT is {PRODUCT#, NAME, PRICE}. The collection of relation schemes used to represent information is called a (relational) database scheme, and the current values of the corresponding relations is called the (relational) database. [Ref. 3: p. 21]

The principal advantage of an RDM is data flexibility. Relationships need not be predefined. We can join PRODUCT tuples with ORDER tuples (Fig. 6), without having to

ORDER relation				PRODUCT relation		
ORD#	DATE	PROD#	QUANT	PROD#	NAME	PRICE
0870	01/16/86	0100	25	0100	CHAIR	25.00
1001	01/21/86	0120	8	0110	TABLE	125.00
1025	01/25/86	0215	10	0120	BOOKCASE	120.00
1236	02/12/86	1025	5	0215	DRESSER	380.00
1142	02/15/86	1132	3	1025	SOFA	289.00
				1100	ARMCHAIR	208.00
				1132	BED	105.00
				1140	COUCH	430.00
				.	.	.
				.	.	.
				.	.	.

Fig. 6. Relational Data Model

predefine the relationships in the design. The RDM can support the use of all types of relationships.

The second advantage is that the way of arranging the data is simple and more understandable to humans than the way of arranging data in the NDM and HDM, since the table structure is simpler than the graph and tree structures.

Another advantage is that query languages provided for relational database processing systems, permit data to be manipulated as groups and not procedurally as one record at a time.

For the above reasons, relational DBMS have become very popular, although it is the youngest of all DBMSs in the computer community.

E. dBASE III CONCEPTS

Recently, database management systems built for microcomputers have become very popular. They provide an inexpensive and easy way for developing database systems for applications like general personnel, accounting, and inventory control.

dBASE III is a relational database management system for microcomputers. It contains its own programming language, permitting a user to develop extremely powerful and complex application programs. dBASE III is used as the DBMS in the design and development of an automated officer assignment database system.

1. Features of dBASE III

- a. Program/data independence. Changes in file structure do not affect application programs.
- b. Data can be easily updated.

- c. Besides the known data types (character, numeric, and logical), it provides the date data type for managing dates, and memo data type for managing long passages of text.
- d. Saves information as disk files in 9 specialized formats each serving a specific dBASE III processing need. [Ref. 5: p. 2-5]
- e. Sorting and indexing capabilities.
- f. Creation and printing of formatted reports.
- g. Date arithmetic.
- h. Built-in high level language, which is extremely powerful and supports structured programming.
- i. Allows interfacing with other software systems, such as SuperCalc, Symphony, WordStar and Basic. [Ref. 4]

2. Limitations of dBASE III

- a. Each database file can have up to 1 billion records maximum. The maximum size of each file is 2 billion bytes.
- b. Allows a maximum of 128 fields in each record with their combined widths up to a maximum of 4,000 characters.
- c. Allows you to have up to 10 database files open at the same time, or 15 files of all types. You can have 7 open index files and 1 format file per active database file.
- d. Filenames can be up to 8 characters long and fieldnames can be up to 10 characters long.
- e. The maximum number of active memory variables is 256. The total number of bytes for memory variables is 6,000.
- f. Execution of dBASE III programs is slower than compiled programs.

All the above values may be limited by the computer hardware configuration. [Ref. 5: p. 2-2]

II. ANALYSIS

Analysis is the study of a problem prior to taking some action. In our case, analysis refers to the study of the existing problem in order to derive the required information which will enable us to decide whether a database system approach can provide an efficient and economical solution to our problem or if it will become part of the problem.

A. PROBLEM DEFINITION

As we stated earlier, the operations required for scheduling the annual assignments of the officers in each Branch of the Hellenic Army General Staff (HAGS) are performed manually by the staff of each Branch. This results in tedious time-consuming operations and inefficiency. In addition, the Branch Director may not be able to make fast decisions concerning personnel management due to the lack of timely and accurate information. Further, the volume of transactions in every big organization and especially in the army, pertaining to personnel management, is getting larger and larger, which means that additional personnel is required to perform the above job, leaving other critical positions unmanned.

For a solution to the above definition of the problem we ask the question 'can a database system provide a more efficient and economical solution?'. In order to answer this question we must be aware of how a Branch is organized, what is being done, how frequently does this job occur, and how great is the volume of transactions.

1. Organization Overview

a. Branch Organization

The Hellenic Army General Staff is organized into three major parts: Arms, Services, and Staff. The relation of these parts to the HAGS organization is summarized in Figure 7.

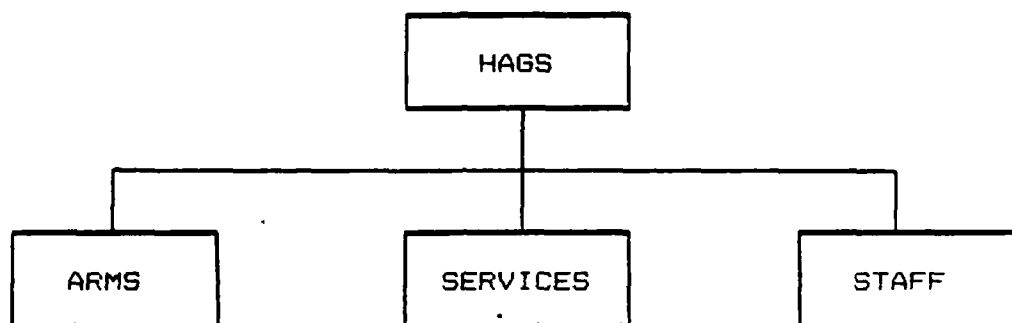


Fig. 7. Organization Chart of the HAGS.

Each Arm or Service (we will call them simply Branches), is responsible for the operational readiness and managerial aspects of the units which are part of the Branch.

Each Branch is organized into subordinate Commands, Staffs, and Units. The number of units varies from Branch to Branch, depending on the mission and special characteristics of each Branch of the Army.

In order to provide a concrete example of a branch organization we will use the Artillery Branch as the model for this research. Figure 8 summarizes the relation of the commands, staffs, and units to the Artillery Branch organization.

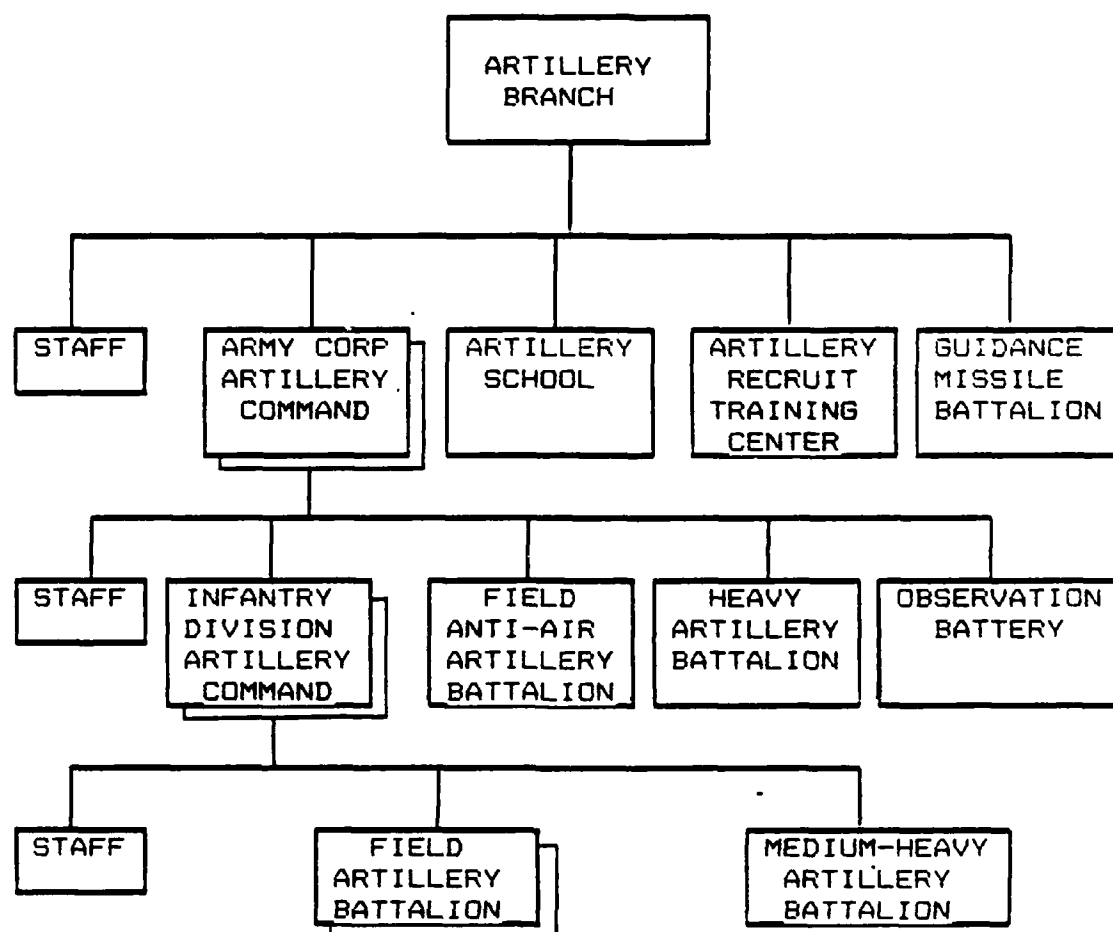


Fig. 8. Organization Chart of the Artillery Branch.

In Table I we provide a detailed list of the Artillery echelons. There are four categories of units: Staffs, Schools, Training Centers, and Combat Units. Also units are characterized as A-type, B-type, or C-type according to their operational readiness. The operational readiness determines the level of manning for each unit. The names, number of units, and the manning level provided below are figurative for security reasons.

Table I. Artillery Echelons

UNIT NAME	UNIT DESCRIPTION	UNIT CATEGORY	UNIT TYPE
ABD/HAGS	Artillery Branch Directorate	Staff	A
AS	Artillery School	School	A
ARTC	Artil. Recruit Training Center	Train. Center	A
GMB	Guidance Missile Battalion	Combat Unit	A
AC/1 AC	1st Army Crop Artil. Command	Staff	A
11 FA/AB	Field Anti-Air Artil. Battal.	Combat Unit	A
12 HAB	Heavy Artillery Battalion	Combat Unit	A
13 OB	Observation Battery	Combat Unit	A
AC/11 ID	11th Inf. Div. Artil. Command	Staff	A
111 FAB	Field Artillery Battalion	Combat Unit	A
112 FAB	Field Artillery Battalion	Combat Unit	A
113 MHAB	Medium-Heavy Artil. Battalion	Combat Unit	A
AC/12 ID	12th Inf. Div. Artil. Command	Staff	B
121 FAB	Field Artillery Battalion	Combat Unit	B
122 FAB	Field Artillery Battalion	Combat Unit	B
123 MHAB	Medium-Heavy Artil. Battalion	Combat Unit	B
AC/2 AC	2nd Army Crop Artil. Command	Staff	B
21 FA/AB	Field Anti-Air Battalion	Combat Unit	C
22 HAB	Heavy Artillery Battalion	Combat Unit	C
23 OB	Observation Battery	Combat Unit	C
AC/21 ID	21st Inf. Div. Artil. Command	Staff	B
211 FAB	Field Artillery Battalion	Combat Unit	B
212 FAB	Field Artillery Battalion	Combat Unit	B
213 MHAB	Medium-Heavy Artil. Battalion	Combat Unit	B
AC/22 ID	22nd Inf. Div. Artil. Command	Staff	C
221 FAB	Field Artillery Battalion	Combat Unit	C
222 FAB	Field Artillery Battalion	Combat Unit	C
223 MHAB	Medium-Heavy Artil. Battalion	Combat Unit	C

b. Officers' Organization

Officers come from two sources, those who have graduated from the Military Academy (MA), and those who used to be warrant officers and have been promoted to officers. They have graduated from the Non-Commissioned Officers School

(NCOS). Further, officers are distinguished according to their specialty as commanding officers (all officers coming from the Military Academy plus a small number coming from NCOS), technicians, and administrative officers (officers coming only from NCOS).

Officers are assigned to various units according to an organization table maintained by each Branch. Table II illustrates the organization table of the Artillery units.

2. Description of the Present Situation

One of the major responsibilities of a Branch is to schedule and monitor the assignments of the officers who belong to this Branch. Each officer during his career has to be assigned to various units and staff positions in order to be equipped with the necessary skills which will qualify him for further professional evolution. For this reason, in each Branch of the HAGS there exist a mechanism through which the Director keeps track of the assignments of his officers.

All officers through the rank of captain are exclusively assigned to units which are part of the corresponding Branch. However, a small number of officers, from the rank of major and on, are unbound by the Branch and disposed to the HAGS to man other staff units outside the Branch. The number of officers serving outside the Branch is always fixed.

Each Branch schedules and monitors the assignments of its officers up to the rank of the Colonel. The assignments of the Generals are scheduled by the HAGS and they will not be discussed here.

Table II. Organization Table of the Artillery Units

		OFFICERS' DISTRIBUTION														
UNIT	Y	06		05		04		03		02		01				SUM
	P	MA	MA	MA	MA	NCOS	MA	MA	NCOS	MA	MA	NCOS	MA	MA	NCOS	
NAME	E	C	C	C	C	T	A	C	C	T	A	C	C	T	A	
ABD/HAGS	A	3	4	2						1						10
AS	A	2	3	2	1	1	1	6	1		6			12	2	39
ARTC	A	1	4	2	1	1	1	9						12	3	37
GMB	A		1	5		1	1	6	1		9				3	30
AC/1 AC	A	1	1	3				1								6
11 FA/AB	A		1	1				3		1	1	3	1		1	12
12 HAB	A		1	1				3			3		1	1		11
13 OB	A		1	1				1			3				1	7
AC/11 ID	A	1	1	2						1						5
111 FAB	A		1	1				3		1	1	2	1		1	12
112 FAB	A		1	1				3		1	1	2	1		1	12
113 MHAB	A		1	1				4		1	1	3	1		1	14
AC/12 ID	B	1		2						1						4
121 FAB	B		1	1				2			2		1	1	1	10
122 FAB	B		1	1				2			2		1	1	1	10
123 MHAB	B		1	1				2		1	1	1	2		1	11
AC/2d AC	B	1	1	2						1						5
21 FA/AB	C		1	1				2			1		1	1		7
22 HAB	C		1	1				2					1	1		7
23 OB	C		1	1						1					1	4
AC/21 ID	B	1		2						1						4
211 FAB	B		1	1				2			2		1	1	1	10
212 FAB	B		1	1				2			2		1	1	1	10
213 MHAB	B		1	1				2		1	1	1	2		1	11
AC/22 ID	C	1		1						1						3
221 FAB	C		1			1		2			1			1	1	8
222 FAB	C		1			1		2			1			1	1	8
223 MHAB	C		1	1				2		1		1		1	1	9
TOTAL		12	31	39	4	3	3	60	6	6	7	48	9	8	8	316

Rank Codes

06=Colonel
05=Lieut. Colonel
04=Major
03=Captain
02=1st Lieutenant
01=2nd Lieutenant

Officers' Origin

MA =Military Academy
NCOS=Non-Commissioned Officers School

Officers' Specialty

C=Commanding
T=Technician
A=Administrative

a. Criteria affecting the Assignments

The mechanism of scheduling the assignments in each Branch of the HAGS is based on certain criteria. The criteria vary from Branch to Branch depending on the organization and special characteristics of each Branch. Among those criteria we will pick up the common ones in order to keep this research more abstract, so that it can be easily applied to every Branch with minor modifications.

- (1) Origin. Officers coming from NCOS can never be assigned to staff positions, as well as in units outside the Branch.
- (2) Specialty. Table II, determines the number of officers assigned to each unit according to their specialty.
- (3) Schools. The only school that can directly affect the assignments is the War College. An officer can never be assigned to a staff unit if he has not graduated from the War College. Other schools that do not affect the assignments are not discussed here, although they may be very important for other aspects in the decision making process.
- (4) Rank. From the Organization Table (Table II) we can determine the number of officers per rank assigned to various units. However, there are some simple rules governing the assignments which cannot be seen in the table and they are stated below.
 - (a) All 2nd lieutenants, right after their graduation from the Military Academy, are assigned to the corresponding Branch-school (Artillery School for our model) for training (specialization). After one year of training, all are assigned to the Artillery Recruit Training Center (ARTC) in order to obtain the necessary training experience. They remain one year in the ARTC and then are assigned to various combat units.
 - (b) All officers graduating from the War College (only majors) are assigned to staff positions (inside or outside the Branch) for at least two years.

- (5) Command Time. Command time is a requirement for all officers up to the rank of lieutenant colonel for promotion to the next rank. The minimum command time required for a lieutenant colonel is one year of service as a Battalion commander. Therefore, each Branch during the scheduling of the assignments must take the command time into consideration, and assign the lieutenant colonels who have not completed this requirement as Battalion commanders. Each commander is responsible to assign appropriate duties to all his subordinate officers, so that they can complete the required command time for their rank.
- (6) Time of Service in the Same Unit. For each rank there is a minimum and a maximum time an officer can serve in the same unit as described below.
- (a) Lieutenants 3-4 years.
 - (b) Captain 3-4 years.
 - (c) Major 1-3 years.
 - (d) Lieutenant colonel 1-3 years
 - (e) Colonel 1-2 years
- (7) Officers' Requests. After an officer has been assigned to a unit for a year he makes a request for his next assignment. This request is submitted only once during each assignment period, except in the case when serious reasons dictate the change of the request and a need for resubmission. In the request each officer states his preferences in three areas that he would like to serve in during his next assignment. The officers' requests are examined by the Branch during the scheduling of the assignments, and in combination with the other criteria. If there is no conflict, all conditions can be satisfied.
- (8) Marital Status. This criterion is examined whenever two or more officers having the same qualifications request the same unit for their next assignment. In this case married officers or officers having bigger families are given preference.
- (9) Historical Data. Each Branch maintains a record for each officer, containing all personal and service data.

All assignments of an officer are maintained in his record, along with the previously mentioned data. This data must always be kept up-to-date, because they reflect the real picture of an officer and provide scheduling personnel with the required information to accomplish their task.

b. Officer Processing Considerations

Officers' assignments are closely related with the promotions. The assignments follow the promotions, which usually occur four times a year: for the colonels (March), for the lieutenant colonels (April), for the majors (May), and for the lieutenants/captains (June). After the announcement of the promotions by the HAGS, each Branch schedules the assignments for the corresponding rank and implements them by issuing the necessary orders.

The personnel involved in this job combines all the above criteria and determines who of the officers meet the requirements for a new assignment and in which unit he must be assigned. Usually one fourth of the officers of each Branch are moved every year during the assignments. Besides this duty, the above personnel are responsible to provide the Branch Director and other staff offices of the HAGS all requested information concerning the officers of the Branch.

The number of personnel involved in this job varies from Branch to Branch, depending on the volume of the officers enrolled in each Branch. In our model usually three people are directly involved, one lieutenant colonel, one lieutenant, and one civilian.

B. JUSTIFICATION OF A COMPUTERIZED SOLUTION

From the above discussion, it is obvious that the manual processing of the officers' assignments is a very tedious, inefficient and time-consuming operation. Three people are working continuously creating, classifying and updating officers' records, scheduling the assignments, and providing the Branch Director and HAGS all requested information (lists and various reports) concerning personnel management. Further, inaccuracies and delays may be introduced in the decision making process due to the great volume of required data and the complex character of the job.

All of the above mentioned problems could be overcome by developing and implementing an automated system.

There are two possible computerized approaches which can provide a solution to our problem: the traditional file processing approach, and a database system approach. Between the two approaches the later is more efficient than the first one for the reasons explained previously. Further, the implementation of a database system on a microcomputer is an economical solution, since the expenses for buying the entire system (hardware and DBMS) are very low (about \$4,000) and they can be offset by a reduction of processing personnel (from three to one). It is apparent that this system would also provide a better quality of services.

From the above discussion it is evident that a database system should be developed and implemented on a microcomputer for an efficient and economical solution to the officers' assignments scheduling problem, as well as, for other problems related with personnel management in each Branch of the HAGS.

C. SYSTEM GOALS AND REQUIREMENTS

In order to establish a framework for the database system development, it is necessary to specify what we expect from the new system, and what capabilities this system must provide.

1. Goals

Goals are targets for achievement. The following targets must be achieved by the system under development:

- a. It should reduce the personnel involved in the process of officers' assignments scheduling by 65 percent.
- b. It should be easy to use by nonprogrammers.
- c. It should be useful.
- d. It should be cost effective.
- e. It should make users' jobs more interesting.

2. Requirements

Requirements specify capabilities that a system must provide in order to solve the problem. Requirements include functional requirements, performance requirements, and requirements for hardware, software, and user interfaces [Ref. 2: p. 33]. The capabilities the new system must provide are the following:

- a. Reliability, i.e. it must be able to perform its intended functions under stated conditions for a stated period of time.
- b. Application development must be easy, cost-effective, and fast.
- c. The data can have multiple uses.

- d. Performance. It must be fully operational 95 percent of each 24-hour period.
- e. Response time to user requests (queries) no more than 5 seconds.
- f. The size of primary memory able to support the system must be at least 320K bytes (180K bytes for bBASE III system program plus 50K bytes for operating system requirements plus 90K for user requirements). [Ref. 5: p. 36]
- g. The computer system must be equipped with a 20M byte hard-disk for the user files and programs, and at least one floppy disk drive for the system program and for back-up purposes.
- h. Maintainability, i.e. software changes must be easy and cost-effective.
- i. Security and privacy.

D. INPUT/OUTPUT INFORMATION

As stated earlier, the system under development can be applied to all Branches of the HAGS with minor modifications, but for the purpose of this research we will include only the Artillery Branch.

Although in this phase we are not able to specify what the exact input and output information will be, we have gained some insights and understanding from the discussion thus far. These thoughts should be taken as hints and guidelines concerning system input and output information for the product design, but not as rigid requirements.

Detailed description of the required input and output information will be provided in the design phase.

1. Input Information

Since the system is intended to deal with officers, the required input information should be officers' data. Therefore, we must consider the following:

- a. Each officer has a unique serial number, rank, origin, specialty, nomination date, promotion date in the current rank, and a home city.
- b. Each officer serves in some unit since a certain date (enrollment date), and has been assigned a duty. The unit is identified by a unique name, has a readiness type, and is located in some geographical area of the country (city and county).
- c. Each officer has a marital status (single, married, divorced, widower, number and age of children, working wife).
- d. Each officer has some education (military/non-military studies).
- e. Each officer submits a request to the Branch indicating three areas he wants to be assigned in his next assignment in preference order.
- f. Each officer has some historical data (previous assignments, duties, promotion dates etc.).

2. Output Information

To meet the above goals and requirements the following output information is required:

- a. List of the scheduled officers' assignments by rank including serial number, name, rank, source unit, destination unit, and date the assignment must take place.
- b. List of any unit including the officers assigned to it, their duties, and enrollment date.
- c. List of all Artillery officers in any requested order.

- d. List of officers by rank reflecting their present status (rank, unit, duties, command time, marital status, and enrollment date).
- e. List of Battalion commanders.
- f. List of all Artillery officers serving outside the Branch.

The above lists will be formatted and issued at any time upon request. Besides these lists, the following reports will be available:

- a. Service time report for any officer including all units he has been assigned to, duties, and enrollment/dis-enrollment dates, in chronological order.
- b. Officer's status report reflecting his status.

III. DESIGN

The design is a solution-the translation of requirements into ways of meeting them [Ref. 6: p. 224]. In our case, database design is the process of developing database structures from those formulated in the analysis phase of Branch requirements. The resulting design must satisfy the needs of the Branch in terms of completeness, integrity, and performance constraints. The design of the system under development includes two steps: the logical (or conceptual) design, and the physical design.

A. LOGICAL (CONCEPTUAL) DESIGN

Logical design is the process of describing the system features, i.e., the functions, inputs, files, the way the files are related to each other in order to form the conceptual database structure, and outputs in a manner that meets the specified requirements. [Ref. 6: p. 225]

1. System Functions

The system under development will perform the following functions:

- a. **Update Operations.** This function allows the user to insert, modify, and delete records in all the supporting files except a few files which will automatically be updated, as they will be described later. It is very important for the Branch to keep all the files up-to-date since the accuracy of the system will mainly depend on the accuracy of the files. Update operations take place whenever a transaction comes to the Branch.

- b. **Assignment Processing Operations.** This function will perform the scheduling of the officers' assignments and will take place right after the annual promotions for each rank. Since the criteria applied in the assignment scheduling are different for each rank, as we described earlier, this function will monitor a number of subsequent functions corresponding one per rank.
- c. **Report Generator.** This function is for retrieving all necessary information from our database upon request. The requested information will be displayed on the screen and optionally sent to the printer.
- d. **Miscellaneous.** This function will include the following:
 - (1) **User Authorization Validation.** Whenever a user attempts to access the database via the existing programs, the system will ask him to enter his password. This password will be checked against the existing list of valid passwords. If it is valid, the system will allow him to use the database. Otherwise an automatic exit to the underlying operating system will take place. In this way we can prevent unauthorized updates and disclosure of the database contents.
 - (2) **User Log.** Every time a valid user enters the system to do a specific task, a record is automatically created containing the user's name, the date and time of database access and the kind of task performed. In this way we can provide an audit trail of who did what on the database and when it took place. In case of erroneous updates it will be easy to find out what exactly happened.
 - (3) **Historical Data Log.** For each transaction concerning officer nomination, promotion, assignment, retirement or death a record is automatically created containing the officer's serial number, rank, type of transaction, unit and date the transaction took place. This file is very useful for historical purposes.

All the above functions will be menu driven. The functional blocks of the system are illustrated in Figure 9.

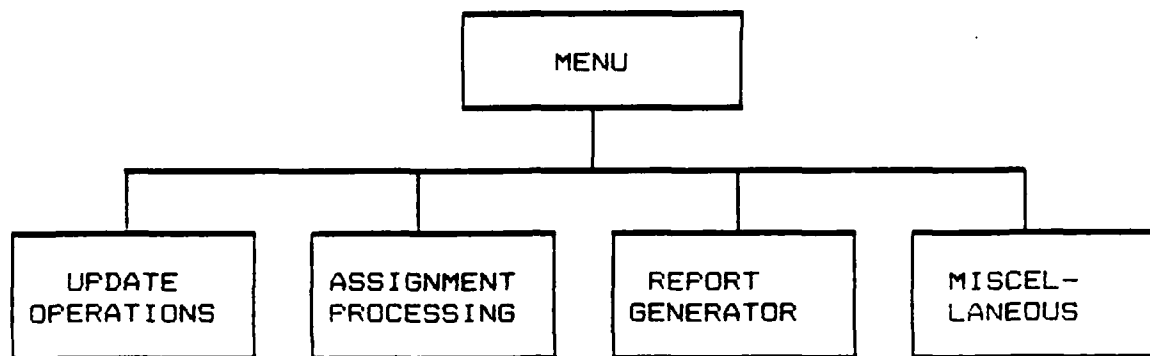


Fig. 9. Functional Blocks of the System.

2. Input Design

During this step we will specify the manner in which data enters the system for processing. In other words we will provide the link that ties our system into the users' world in a way that guarantees the reliability of the system, avoids extra steps and delays, and keeps the process simple. [Ref. 6: p. 286]

The most efficient way to achieve the above objectives is to design a menu-driven on-line system. A menu is a screen of information displayed on the CRT that shows the user what functions can be performed and how to select them.

A main menu and a number of sub-menus will guide the user of our system to select and perform the appropriate functions in a top-down fashion as described below.

a. Main Menu Description

The main menu in Figure 10 shows the options available to the user of the database system. Each option is

identified by a number. To invoke a particular option, the user depresses the key corresponding to the desired option.

```

*  MAIN MENU  *
DATABASE UPDATE:..... 1
ASSIGNMENT PROCESSING:... 2
REPORT GENERATOR:..... 3
END OF SESSION:..... 4
EXIT TO DOS:..... 5
Enter your selection ->

```

Fig. 10. Main Menu.

b. Sub-menus Description

(1) Update Menu. This menu is entered when a user selects option 1 from the main menu. As we can see in Figure 11, it provides seven new options. Through this menu we can insert, modify, delete records in the specified database

```

*  UPDATE MENU  *
INSERT RECORDS INTO OFFICER FILE:..... 1
INSERT RECORDS INTO SCHOOLS FILE:..... 2
INSERT RECORDS INTO HISTORIC FILE:..... 3
MODIFY RECORDS FROM OFFICER FILE:..... 4
MODIFY RECORDS FROM REQUEST FILE:..... 5
DELETE RECORDS FROM OFFICER FILE:..... 6
EXIT TO MAIN MENU:..... 7
Enter your selection ->

```

Fig. 11. Update Menu.

files, or exit to main menu. The selection of the desired records is done by typing the record key of the corresponding database file. After selecting a specific record, its structure is displayed on the screen and the cursor is positioned in the first element to be updated. Invalid data types (numeric, character, logical, or date) are not accepted by the system. Thus, the user is protected from typing invalid data types.

(2) Assignment Processing Menu. This menu is displayed in case a user selects option 2 from the main menu (Figure 10). This menu provides seven options as it is shown in Figure 12. Each function is performed by selecting the corresponding number. The actual input data required for processing the assignments are database files, but those files are selected by the corresponding application programs, so the user does not have to worry.

```

*  ASSIGNMENT PROCESSING MENU  *
1st LIEUTENANT ASSIGNMENT PROCESSING:... 1
2nd LIEUTENANT ASSIGNMENT PROCESSING:... 2
CAPTAIN ASSIGNMENT PROCESSING:..... 3
MAJOR ASSIGNMENT PROCESSING:..... 4
LIEUT. COLONEL ASSIGNMENT PROCESSING:... 5
COLONEL ASSIGNMENT PROCESSING:..... 6
EXIT TO MAIN MENU:..... 7
      Enter your selection ->

```

Fig. 12. Assignment Processing Menu.

(3) Report Generator Menu. This menu is displayed when the user selects option 3 from the main menu (Figure 10).

From this menu a user can select the desired output (list or report). Figure 13 shows the details of this menu.

* <u>REPORT GENERATOR MENU</u> *	
LIST OF SCHEDULED ASSIGNMENTS:.....	1
LIST OF OFFICERS OF ANY DESIRED UNIT:.....	2
LIST OF OFFICERS IN ANY DESIRED ORDER:.....	3
LIST OF OFFICERS OF ANY DESIRED RANK:.....	4
LIST OF BATTALION COMMANDERS:.....	5
LIST OF OFFICERS SERVING OUTSIDE THE BRANCH:	6
OFFICER'S SERVICE TIME REPORT:.....	7
OFFICER'S STATUS REPORT:.....	8
EXIT TO MAIN MENU:.....	9
Enter your selection ->	

Fig. 13. Report Generator Menu.

3. File Design

To support the above specified functions the following files with the corresponding structures will be created. The names of the files and fields are the ones that will be used in our system.

a. OFFICER File.

It is the main file containing all required information for each officer. Figure 14 shows the structure of the OFFICER file, as well as the explanation of the fields.

<u>FIELD</u>	<u>NAME</u>	<u>TYPE</u>	<u>WIDTH</u>	<u>FIELD-EXPLANATION</u>
01	NAME	C	018	Officer's name
02	<u>SERNO</u>	C	005	Officer's serial number
03	RANK	C	002	Officer's rank
04	NOMYEAR	C	002	Year of nomination
05	SPECIALTY	C	001	Officer's specialty
06	SOURCE	C	004	Officer's source (MA,NCOS)
07	NOMDATE	Date	008	Nomination Date
08	PROMDATE	Date	008	Promotion Date
09	ASWEIGHT	C	002	Weight for next assignment
10	ORIGCITY	C	006	City of officer's origin
11	ORIGCOUNTY	C	008	County of officer's origin
12	MARSTAT	C	001	Officer's marital status
13	CHILDREN	N	001	Number of children
14	WORKWIFE	L	001	Working wife

Primary Key : SERNO

Fig. 14. Structure for file OFFICER.

b. SERVES File

This file contains information reflecting the current service status of each officer (unit he is assigned to, enrollment date, duty). Its structure and the explanation of fields is shown in Figure 15.

<u>FIELD</u>	<u>NAME</u>	<u>TYPE</u>	<u>WIDTH</u>	<u>FIELD-EXPLANATION</u>
01	<u>SERNO</u>	C	005	Officer's serial number
02	<u>UNITNAME</u>	C	008	Unit name
03	<u>ENROLDATE</u>	DATE	008	Enrollment date
04	DUTY	C	010	Officer's duty

Primary Key : {SERNO, UNITNAME}

Fig. 15. Structure for File SERVES.

c. REQUESTS File

This file contains officers' requests for their next assignment. Figure 16 provides the details of this file.

<u>FIELD</u>	<u>NAME</u>	<u>TYPE</u>	<u>WIDTH</u>	<u>FIELD-EXPLANATION</u>
01	SERNO	C	005	Officer's serial number
02	STAFFOFFIC	L	001	Staff Officer (T or F)
03	SUBMDATE	DATE	008	Request's submission date
04	UNIT1	C	008	First requested unit
05	UNIT2	C	008	Second requested unit
06	UNIT3	C	008	Third requested unit

Primary Key : SERNO

Fig. 16. Structure for File REQUESTS.

d. ASSIGNED File

This is a temporary file created during the assignment processing, containing information concerning the officers to be assigned to some unit. Details of this file are provided in Figure 17.

<u>FIELD</u>	<u>NAME</u>	<u>TYPE</u>	<u>WIDTH</u>	<u>FIELD-EXPLANATION</u>
01	SERNO	C	005	Officer's serial number
02	RANK	C	002	Rank
03	SOURCE	C	004	Source (MA or NCOS)
04	SPECIALTY	C	001	Specialty
05	UNITNAME	C	008	Unit name
06	ASGNDATE	DATE	008	Assignment date
07	ASNWEIGHT	N	002	Assignment Weight

Primary Key: {SERNO, UNITNAME}

Fig. 17. Structure for File ASSIGNED.

e. UNITORG File

This file contains information about the organization of each unit according to the operational readiness type and type of echelon as depicted in Figure 18.

<u>FIELD</u>	<u>NAME</u>	<u>TYPE</u>	<u>WIDTH</u>	<u>FIELD-EXPLANATION</u>
01	ECHELON	C	005	Type of echelon
02	<u>READINESS</u>	C	001	Operational readiness
03	MAC06	N	002	MA commanding colonels
04	MAC05	N	002	MA commanding lt. colonels
05	MAC04	N	002	MA commanding majors
06	NCOSC04	N	002	NCOS commanding majors
07	NCOST04	N	002	NCOS technician majors
08	NCOSA04	N	002	NCOS administrative majors
09	MAC03	N	002	MA commanding captains
10	NCOSC03	N	002	NCOS commanding captains
11	NCOST03	N	002	NCOS technician captains
12	NCOSA03	N	002	NCOS administrative captains
13	MAC02	N	002	MA commanding 1st lieuten.
14	NCOSC02	N	002	NCOS commanding 1st lieuten.
15	NCOST02	N	002	NCOS technician 1st lieuten.
16	NCOSA02	N	002	NCOS admin. 1st lieutenants
17	MAC01	N	002	MA commanding 2nd lieuten.
18	NCOSC01	N	002	NCOS commanding 2nd lieuten.
19	NCOST01	N	002	NCOS technician 2nd lieuten.
20	NCOSA01	N	002	NCOS admin. 2nd lieutenants

Primary Key: {ECHELON, READINESS}

Fig. 18. Structure for File UNITORG.

f. UNIT File

This file contains information about each unit. The Structure of the file and explanation of fields is depicted in Figure 19.

<u>FIELD</u>	<u>NAME</u>	<u>TYPE</u>	<u>WIDTH</u>	<u>FIELD-EXPLANATION</u>
01	<u>UNITNAME</u>	C	008	Unit name
02	<u>CATEGORY</u>	C	001	Unit category
03	<u>ECHOLON</u>	C	005	Type of echelon
04	<u>READINESS</u>	C	001	Unit readiness
05	<u>CITY</u>	C	010	City of unit station
06	<u>COUNTY</u>	C	010	County of unit station

Primary Key: UNITDESCR

Fig. 19. Structure for File UNIT.

h. HISTORIC File

This file records all major officer's transactions which have taken place during his career. Major transactions are considered to be the nomination, promotions, assignments, retirement, and death. Usually there are more than one entry for each officer in this file. The structure of the file and field explanation is shown in Figure 20.

<u>FIELD</u>	<u>NAME</u>	<u>TYPE</u>	<u>WIDTH</u>	<u>FIELD-EXPLANATION</u>
01	<u>SERNO</u>	C	005	Officer's serial number
02	<u>RANK</u>	C	002	Officer's rank
03	<u>TRANSTYPE</u>	C	012	Transaction type
04	<u>UNIT</u>	C	008	Unit name
05	<u>TRANSDATE</u>	DATE	008	Date the transaction occurred
06	<u>ORDERID</u>	C	020	Order caused the transaction

Primary Key: {SERNO, TRANSDATE}

Fig. 20. Structure for File HISTORIC

f. SCHOOLS File

This file contains information about military and non-military studies of the officers. It is possible for an officer to have more than one record in this file depending on the number of schools he has attended. Records are created only for officers who studied for at least one year in some school. Details are shown in Figure 21.

<u>FIELD</u>	<u>NAME</u>	<u>TYPE</u>	<u>WIDTH</u>	<u>FIELD-EXPLANATION</u>
01	SERNO	C	005	Officer's serial number
02	<u>SCHOOLNAME</u>	C	010	School-name
03	<u>DEGREE</u>	C	012	Title obtained
04	OBJECT	C	018	Object of studies
05	COUNTRY	C	010	Country of studies
06	DURATION	N	002	Studies duration in months
07	GRADDATE	DATE	008	Graduation date

Primary Key : {SERNO,SCHOOLNAME}

Fig. 21. Structure of file SCHOOLS.

i. SELECTED file

This file contains all officers who have been selected by the Branch to study in some school for at least one year. The structure of the file is shown in Figure 22.

<u>FIELD</u>	<u>NAME</u>	<u>TYPE</u>	<u>WIDTH</u>	<u>FIELD-EXPLANATION</u>
01	SERNO	C	005	Officer's serial number
02	SCHOOLNAME	C	010	Name of the school
03	SCHOOLYEAR	C	002	Year the officer will be sent to the school

Primary Key: {SERNO}

Fig. 22. Structure for File SELECTED

j. USERLOG File

This file records all users' activities on the database system. Details of the file structure and field-explanation are shown in Figure 23.

<u>FIELD</u>	<u>NAME</u>	<u>TYPE</u>	<u>WIDTH</u>	<u>FIELD-EXPLANATION</u>
01	USERNAME	C	018	User's name used the system
02	TASK	C	010	Task performed in the system
03	PROGRAM	C	10	Program executed
03	<u>LOGDATE</u>	DATE	008	Date of using the system
04	<u>LOGTIME</u>	C	005	Time of using the system

Primary Key: {LOGDATE, LOGTIME}

Fig. 23. Structure for File USERLOG.

i. USERID File

This file contains all valid passwords and the user's name corresponding to each password, as it is described in Figure 24.

<u>FIELD</u>	<u>NAME</u>	<u>TYPE</u>	<u>WIDTH</u>	<u>FIELD-EXPLANATION</u>
01	PASSWORD	C	006	A predefined password
02	USERNAME	C	018	User's name

Primary Key: PASSWORD

Fig. 24. Structure for File USERID.

4. Conceptual Database Structure

The files described above are related as illustrated in the entity/relationship diagram shown in Fig. 25. The rectangles represent entity sets (files), and the diamonds represent relationships between entity sets. A relationship is also an entity set containing attributes (usually the keys) from all other entity sets which are linked together via this relationship. As we can see in this diagram each officer serves in some unit which has an organization, he requests some units to be assigned to in his next assignment, he may be assigned to some unit, he may have studied in some school(s) or he may have been selected to study in some school, and he has some historic data. The entity sets USERID and USERLOG are not linked with the other entity sets in the diagram since their function is independent from the other ones.

Finally, the relational database scheme is presented in Figure 26.

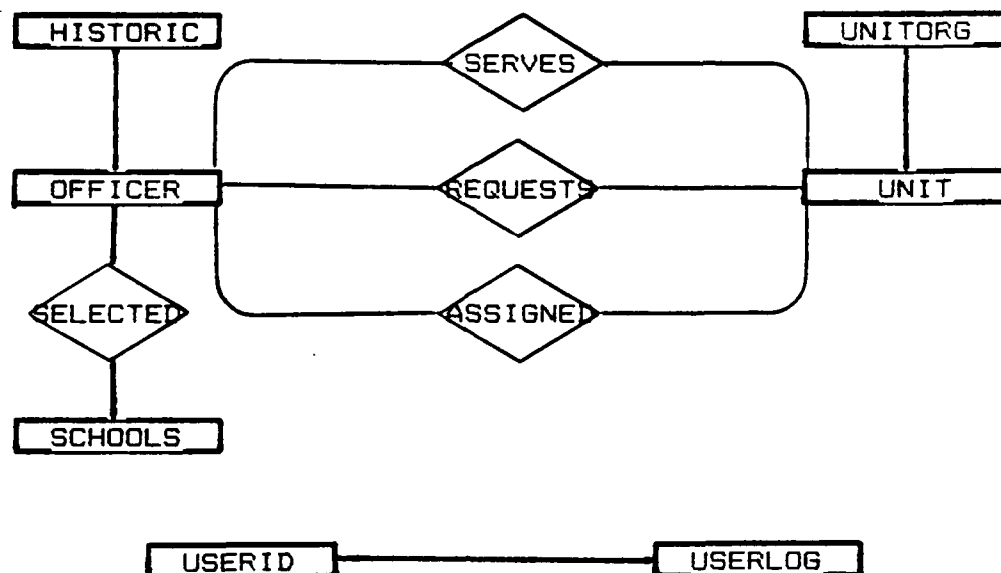


Fig. 25. Entity/Relationship Diagram.

OFFICER {NAME, SERNO, RANK, NOMYEAR, SPECIALTY, SOURCE, NOMDATE, PROMDATE, ASNWEIGHT, ORIGCITY, ORIGCOUNTY, MARSTAT, CHILDREN, WORKWIFE}

SERVES {SERNO, UNITNAME, ENROLDATE, DUTY}

REQUESTS {SERNO, WCFINISHED, SUBMDATE, UNIT1, UNIT2, UNIT3}

ASSIGNED {SERNO, RANK, SOURCE, SPECIALTY, UNITNAME, ASGNDATE, ASNWEIGHT}

UNIT {UNITNAME, CATEGORY, ECHELON, READINES, CITY, COUNTY}

UNITORG {ECHELON, READINESS, MAC06, MAC05, MAC04, NCOSCO4, NCOST04, NCOSA04, MAC03, NCOSCO3, NCOST03, NCOSA03, MAC02, NCOSCO2, NCOST02, NCOSA02, MAC01, NCOSCO1, NCOST01, NCOSA01}

SELECTED {SERNO, SCHOOLNAME, SCHOOLYEAR}

SCHOOLS {SERNO, SCHOOLNAME, DEGREE, OBJECT, COUNTRY, DURATION, GRADDATE}

HISTORIC {SERNO, RANK, TRANSType, UNIT, TRANSDATE, ORDERID}

USERID {PASSWORD, USERNAME}

USERLOG {USERNAME, TASK, PROGNAME, LOGDATE, LOGTIME}

Fig. 26. The Relational Database Scheme.

5. Output Design

The most important feature of an information system for users is the output it produces. No matter how reliable a system is, if it does not produce quality output, users may feel the entire system unnecessary. [Ref. 6: p. 231]

The term 'output' is applied to any information produced by a system whether printed, displayed, or spoken. In our case the produced output will be printed and/or displayed, depending on how the user wants it. Our goal is to design the output in such a way that it can be easily read and understood by the user.

As discussed earlier the system output will be lists and reports (Figure 13). The length of the output lines will be up to 80 characters in order to be able to fit either on the screen or on the standard 8"x11" sheet of paper.

a. Lists

(1) List of Scheduled Assignments. This list is produced each time assignment processing function is executed. The format of this list is shown in Figure 27.

ABD/HAGS

DATE:.....

LIST OF SCHEDULED ASSIGNMENTS FOR ... (RANK)....

SERIAL:	NAME	UNIT		DUE
NUMBER:		SOURCE	DESTINATION	DATE
20793	Armstrong David K.	AC/1 AC	AS	05/21/86
22467	Babbitt Almon P.	AS	ABD/HAGS	05/25/86
20845	Norton Harold G.	ABD/HAGS	111 FAB	05/14/86
.
.
.

Fig. 27. List of Scheduled Assignments.

(2) List of Officers in Any Desired Order. This List (Figure 28) contains all Artillery officers in any of the following orders:

- (a) Alphabetical.
- (b) Rank.
- (c) Specialty.
- (d) Rank and alphabetical.

ABD/HAGS

DATE:../../..

LIST OF ARTILLERY OFFICERS IN ORDER

RANK	NAME	SERIAL	SOU-	SPECI-	UNIT	MARITAL
		NUMBER	RCE	ALTY		STATUS
08	Down Kenneth R.	20001	MA	C	ABD/HAGS	M
07	Calaunan Tend G.	20017	MA	C	ARTC	M
:	:	:	:	:	:	:
:	:	:	:	:	:	:

Fig. 28. List of Artillery Officers in Some Order.

(3) List of Officers of a Desired Unit. This list contains all officers serving in a specific unit. The format of the list is shown in Figure 29.

ABD/HAGS

DATE:../../..

LIST OF OFFICERS SERVING IN ... (UNIT)...

RANK	SERIAL	NAME	DUTIES	ENROLLMENT
	NUMBER			DATE
05	20261	Franklin Adams P.	Battalion CDR	05/20/84
04	20768	Ervin Joseph H.	Deputy CDR	06/25/85
.

Fig. 29. List of Officers of a Specific Unit.

(4) List of Officers of Any Desired Rank. This list contains all Artillery officers of any desired rank. The format of the list is shown in Figure 30.

ABD/HAGS

DATE:../../..

LIST OF ARTILLERY ... (RANK)...

SERIAL NUMBER	NAME	SOURCE	SPECIALTY	UNIT	MARITAL STATUS
20270	Bell Richard K.	MA	C	111 FAB	M
20273	Anzini Danniell D.	MA	C	113 MHAB	M
20275	Ariss Bruce F.	MA	C	ABD/HAGS	S
20277	Gray Joseph W.	MA	C	AC/1 AC	D
.
.
.

Fig. 30. List of Officers of Any Desired Rank.

(5) List of Battalion Commanders. This List contains all Artillery Battalion commanders. Details of the list are provided in Figure 31.

ABD/HAGS

DATE:../../..

LIST OF ARTILLERY BATTALION COMMANDERS

RANK	NAME	UNIT	ENROLLMENT DATE
05	Cabral David T.	212 FAB	06/26/85
05	Gray Joseph W.	112 FAB	06/18/84
05	Franclin Adams P.	211 FAB	06/23/85
05	Norton Harold G.	111 FAB	07/01/84
05	Jarecki Edward L	11 FA/AB	07/28/85
.	.	.	.
.	.	.	.
.	.	.	.

Fig. 31. List of Battalion Commanders.

(6) List of Officers Serving Outside the Branch.

This list contains all Artillery officers who serve outside the Branch. The format of the list is shown in Figure 32.

ABD/HAGS

DATE:../../..

LIST OF ARTILLERY OFFICERS SERVING OUTSIDE THE BRANCH

<u>RANK</u>	<u>NAME</u>	<u>UNIT</u>	<u>ENROLLMENT DATE</u>
07	Billeb James W.	HAGS	03/12/85
06	Wapper Alfrend D.	1 AC	04/20/85
.	.	.	.
.	.	.	.
.	.	.	.

Fig. 32. List of Officers Serving Outside the Branch.

b. Reports

(1) Officer's Service Time Report. This report contains a summary of the major transactions of an officer, having occurred during his career. The format of this report is shown in Figure 33.

ABD/HAGS

DATE:../../..

OFFICER'S SERVICE TIME REPORT

SERIAL NUMBER: NAME:
SOURCE: SPECIALTY: ... ORIGIN:.....

<u>DATE</u>	<u>TRANSACTION TYPE</u>	<u>RANK</u>	<u>UNIT</u>	<u>ORDER ID</u>
07/20/60	Nomination	01	MA	F. 430/21/621/NDD
08/20/60	Assignment	01	AS	F. 435/24/321/ABD
.
.

Fig. 33. Officer's Service Time Report.

(2) Officer's Status Report. This report provides all information reflecting an officer's current status. The format of this report is shown in Figure 34.

ABD/HAGS

DATE:../../..

OFFICER'S STATUS REPORT

NAME	:
SERIAL NUMBER	:
RANK	:
SOURCE	:
SPECIALTY	:
NOMINATION YEAR	:	..
MARITAL STATUS	:
CHILDREN	:	.
ORIGIN	:
UNIT	:
ENROLLMENT DATE	:	../../..
DUTIES	:

Fig. 34. Officer's Status Report.

B. PHYSICAL DESIGN

Physical design is the process of transformation. The logical schema is transformed into a working system. This transformation is done through the tools that are available with the DBMS to be used. [Ref. 1: p. 188] More specifically, during this step we will define the files described above and store them into our database, as well as create the programs that will manipulate the files through the dBASE III DBMS.

The database schema or more simply the files that form the database are defined via the Data Definition Language (DDL) provided by the DBMS, and the programs are created via the Data Manipulation Language (DML). dBASE III provides only one language which serves both purposes.

1. File Definition/Creation

The files are defined by using the 'CREATE' dBASE III command. This command allows us to define the structure of each file to be used by the database system. Since a file is a collection of records of the same type, our job here is to describe the structure of the records of each file, i.e., the attribute (field) names, their types (character, numeric, logical, date), and their sizes. Using this command we define all files described above.

The records of each file are stored in the database by using the command 'APPEND'. Before we use this command we have to open the database file the records to be stored in by using the command 'USE fn', where 'fn' is the file name. The records are stored sequentially in the database file.

A small number of records for each file has been created and stored in the database, so that we will be able to test the programs when they have been completed.

2. Program Creation

The next step is to write the software which will perform the specified functions by manipulating our database via the dBASE III DBMS, and producing the required output.

Our primary goal is to produce a comprehensive working system that can be effectively and easily used by people who are not programmers.

The whole program has been decomposed into a number of interconnected modules in a hierarchical top-down fashion, as depicted in the program structure chart in Figure 35. The actual programs are presented in APPENDIX A.

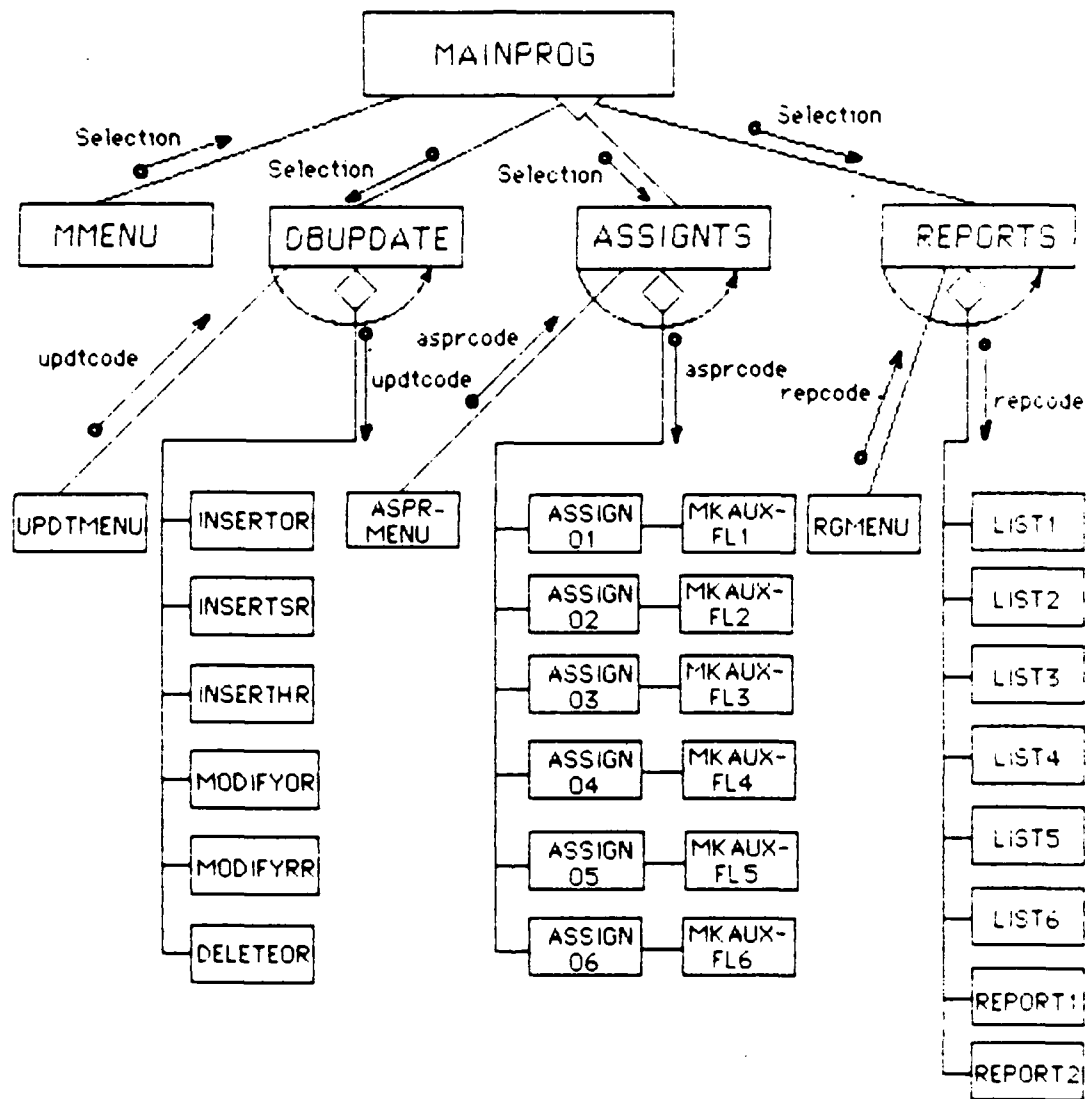


Fig. 35. Program Structure Chart

IV. SYSTEM IMPLEMENTATION

Implementation includes all those activities that take place to convert from the old system to the new. The new automated system which has been already developed, is proposed to replace the existing manual system. Proper implementation is essential to provide a reliable system to meet the specified requirements.

The aspects of implementation which will be discussed here include hardware requirements, training personnel, conversion procedures, and software description/documentation.

A. HARDWARE REQUIREMENTS

To implement our system using dBASE III DBMS we need a 16-bit microcomputer using MS DOS or PC DOS version 2.0 or later. Any 16-bit microcomputer that is fully software compatible with the the IBM PC or IBM PC/XT should be able to use the system. The computer must have a minimum of 320K of RAM, one 20M byte hard-disk drive, and at least one floppy disk drive. Any printer that can print 80 columns of text can be used.

The RAM is allocated as follows.

1. About 50K bytes for operating system requirements (resident part of DOS).
2. 180K bytes for the dBASE III system program.
3. About 16K bytes for the user's program currently being executed (this is the size of the biggest program).

4. The rest of memory is allocated to the required working areas (buffers for the files currently being open).

In case we want to use another customer-supplied program (word processor, etc.) along with dBASE III, additional memory is required.

The hard-disk size has been calculated on the basis of a full scale-model as follows.

1. About 130K bytes for the user programs.
2. About 12M bytes for the required database files and index files. These files will contain data for 10,000 officers (worst case senario).
3. About 12K bytes for the format files.
4. About 100K for text files.

The rest of the hard-disk space can be used for future improvements of the system, as well as for other data storage needs.

B. TRAINING PERSONNEL

Since the system is implemented on a microcomputer, the need for training is limited only to a few individuals who will be both operators and users. Those individuals who are going to use the new system should know how to use a microcomputer (how to turn on the system, how to insert a diskette, when it is safe to turn off the equipment without danger of data loss, or how to determine whether a problem arising is caused by the equipment, software or something they have done in using the system), what software they will use for the system, and how they should use it to perform a specific task.

As the above discussion demonstrates, there are two aspects to user training: familiarization with the processing

system itself (equipment used) and training in using the the application (that is, the software that accepts the data, processes it, and produces the results). Our intent is to train a few users how to use the system in order to accomplish their task, and not to teach them how to write new programs. Therefore, both aspects mentioned above can be covered in a very short time period (one or two weeks).

C. CONVERSION PROCEDURES

Conversion is the process of changing the old system to the new one. The method we will use for converting from the existing manual to the new automated system is the **parallel systems**. That is, the Branch will continue to operate the old system in the accustomed manner but it also will begin using the new system. This method is the safest conversion approach, since it guarantees that if possible errors will arise in using the new system (processing errors, inability to handle certain types of transactions, or other inefficiencies) they will not harm the Branch, since the old system is still in use. Another important advantage of the parallel systems approach is that before the Branch abandons the manual system, there will be a trial period for the new system during which revealed errors and inefficiencies can be eliminated without loss of time, revenue, or service.[Ref. 6: p. 532]

D. SOFTWARE DESCRIPTION/DOCUMENTATION

The programs which implement our database system have been grouped into six categories according to the function they perform: main program, main-menu and sub-menu programs,

programs supporting the update operations, programs performing the assignment processing, programs producing the required lists and reports, and miscellaneous programs. All programs (code), as well as lists and reports produced by the programs are presented in APPENDIX A.

1. Main Program

This program controls the entire operation of the system and it is the only program that the user calls by name (MAINPROG). After initializing basic dBASE III functions, MAINPROG checks if the user is authorized to use the database system by prompting him to enter his password. Unauthorized users are aborted and the program exits automatically to the operating system (DOS), displaying the appropriate message.

In case the user is an authorized one, MAINPROG calls program GRFLAG which displays on the screen the Greek Flag. After a small delay it calls program DBSTITLE which provides the title of the database system and prompts the user to hit any key to continue. Then program MMENU is called which is the main-menu of the system, and pauses waiting for the user to make his choice. Then a CASE statement permits the program to branch to the appropriate program group according to the users choice or exit either to dBASE III or to the underlying operating system. A DO WHILE loop causes the process to be repeated until the user selects option 4 or 5 in which case the program terminates and control passes to dBASE III or to the operating system, respectively.

2. Main-Menu and Sub-menu Programs

a. Main-Menu Program (MMENU)

This program displays on the screen the main-menu. This menu provides a screen of information that shows the user what functions can be performed and how to select them. Those functions correspond to update operations, assignment processing, and report generation. In addition it provides the option to exit to either dBASE III or the operating system. A flashing label at the top of the menu informs the user that the main-menu is on the screen, and a highlighted label at the bottom prompts him to enter his choice. The user's choice is expressed by one of the numeric characters 1 through 9. A 'RANGE' statement checks each time the user types a character and in case it lies outside the specified range an appropriate message is displayed. Also the 'PICTURE '9'' statement guarantees that the program does not accept alphabetical or other characters. The subsequent sub-menus described below work in a similar fashion.

b. Sub-menu Programs

Three sub-menus corresponding one per basic function of the system (update operations, assignment processing, report generation) guide the user to perform the appropriate task. Those sub-menus are created by the following programs.

(1) UPDTMENU. This program provides a screen of information concerning the update operations. Via this program the user can insert, modify, or delete records from the files whose names are presented on the screen. UPDTMENU is called by the DBUPDATE program in case the user selects option 1 from the main-menu.

(2) ASPRMENU. This program displays information concerning the assignment processing operations. Through this program the user can access any of the programs which perform the officers' assignments. ASPRMENU is called by the ASSIGNTS program in case the user selects option 2 from the main-menu.

(3) RGMENU. This program provides a screen of information concerning the report generation operations. From this submenu the user can access any of the programs which produce the required lists and reports. RGMENU is called by the REPORTS program in case the user selects option 3 from the main-menu.

3. Programs Supporting the Update Operations

This group of programs performs all the update operations. That is, using these programs we can insert new records into the database files, as well as modify or delete existing records from the database files. The programs which implement those operations are the following:

a. DBUPDATE

This program controls the entire update operation and is executed whenever the user selects option 1 from the main-menu. First the program calls the UPDTMENU program, which is the update sub-menu we described above, and pauses waiting for the user to make his choice which is stored in the variable 'updtcode'. Then a CASE statement permits the program to branch to the corresponding update program or exit to main-menu. A DO WHILE loop allows the program to keep running until the user decides to exit to main-menu.

b. INSERTOR

This program allows the user to insert new records into the OFFICER file which is the main database file. It is called by the DBUPDATE program in case the user selects option 1 from the update sub-menu (UPDTMENU program).

The program calls the MFRAME program which displays a window on the left half of the screen into which all program messages and user data are placed during the execution of the program. A message is displayed on the screen prompting the user to enter the serial number of the officer to be inserted into the file. Then the program searches the OFFICER file using as key the officer's serial number to see whether the record exists in the database or not. If the search is successful, a message is displayed informing the user that the record already exists, and he is asked if there are more records to be inserted into the database. In case the search operation is unsuccessful (that is, record does not exist in the database) the user is asked if he needs the codes required for the insert operation (rank, source, specialty, and marital status codes). If his answer is 'Y' (yes) the program WINDOW3 is called which displays a window occupying the right half of the screen containing all codes and their explanation. In this way the user can have on-line help and the risk of incorrect data input is reduced. Next the format of the record to be inserted is displayed on the left frame and the cursor is positioned at the first field.

When the user finishes the input of data for the particular officer, the record is appended to the OFFICER file and the necessary entries are created in some other database files. Namely a record is created into the HISTORIC file containing the officer's serial number, the nomination date, the order by which the nomination of the officer has been

known, and the unit in which he serves. The unit name and the order identification are entered by the user. Then the created record is displayed on the screen. Also a record is created into the REQUESTS file containing only the officer's serial number. All other fields remain empty. In case the source of the officer is 'NCOS' (Non-Commissioned Officers School), a record is created into the SERVES file containing his serial number the unit he serves and the enrollment date.

A DO WHILE loop keeps the program running as long as the user's answer to the prompt 'MORE INSERTIONS? (Y/N)' is 'Y'.

c. INSERTSR

This program permits the user to insert new records into the SCHOOLS file. No other files are updated during the execution of this program. The program is called by the DBUPDATE program in case the user selects option 2 from the update sub-menu.

An officer can have more than one record in the SCHOOL file depending on what and how many schools he has graduated from. The process goes in a similar way with the described above (INSERTOR program) except that the search is based on the compound key consisting of the fields serial number and school name ({SERNO,SCHOOLNAME}). The user's answer 'N' to the program's prompt 'MORE INSERTIONS? (Y/N)' causes the program to be terminated.

d. INSERTHR

Through this program the user can insert new records into the HISTORIC file. It is called by the DBUPDATE program whenever the user selects option 3 from the update

sub-menu. The only legal transactions which can cause new records to be inserted into the HISTORIC file from this program are those concerning retirement or death. All other transactions are aborted by the program. The process is similar with that described above.

e. MODIFYOR

This program is called by the DBUPDATE program in case the user selects option 4 from the update sub-menu, and allows him to modify records into the OFFICER file.

When the program is entered, it calls the MFRAME program whose purpose has been explained during the INSERTOR program description. Then a message prompts the user to enter the serial number of the officer whose record is to be modified. The program searches the OFFICER file using as the key the serial number typed by the user. In case of an unsuccessful search an appropriate message is displayed and after a small delay the user is asked if he wants to perform more modifications. If his answer is 'Y' (yes) the process is repeated. In the case of a successful search a menu is displayed inside the frame created by the MFRAME program that shows the user which of the record-fields can be modified and prompting him to enter his selection. The fields which can be modified in the OFFICER file are the 'NAME', 'RANK' and 'PROMDATE' (promotion date), and those concerning the family status ('MARSTAT', 'CHILDREN', 'WORKWIFE'). After the user's response, the structure of the fields to be modified are displayed on the screen (inside the frame) and the cursor is positioned at the first field. After the displayed fields have been modified by the user, the modified record is displayed in an appropriate format.

In case the rank status has been changed, a record in the HISTORIC file is automatically created including the new transaction (PROMOTION). The new record is displayed on the right half of the screen in a formatted way.

The process keeps going until the user's answer to the program's prompt 'MORE MDIFICATIONS? (Y/N)' is 'N'.

f. MODIFYRR

This program allows the user to modify records in the REQUESTS file, and it is called by the DBUPDATE program in case the user selects option 5 from the update sub-menu. All fields of the records under modification can be changed except the serial number which is the primary key. The process is similar with that described above, except that it is simpler since no other files are updated during the execution of this program. The program terminates when the user's answer to the program's prompt 'MORE MODIFICATIONS? (Y/N)' is 'N'.

g. DELETEOR

Through this program we can delete records from the OFFICER file. It is called by the DBUPDATE program whenever the user selects option 6 from the update sub-menu. The process goes as follows.

The program calls the MFRAME program (described earlier) and prompts the user to type the serial number of the officer to be deleted. Then using the serial number as the key searches the OFFICER file to find the corresponding record. In case of an unsuccessful search an appropriate message is displayed and the user is asked if he wants to continue with more deletions. If his answer is 'Y' the process is repeated. In case of a successful search, the record is displayed on the

screen along with the prompt 'DELETE? (Y/N)'. In this way the user has the chance to prevent accidental deletions of records. If the user's answer is 'Y' the record is marked for deletion. Then the program searches the files HISTORIC, SERVES, REQUESTS, and SCHOOLS with the same key, and all records that match with the key are marked for deletion.

The process keeps going until the user's answer to the program's prompt 'MORE DELETIONS? (Y/N)' is 'N'. In this case, all files which include records marked for deletion are packed, and control passes to the DBUPDATE program.

File packing is a time-consuming operation since the entire file is searched and marked for deletion records are removed. The situation is even worst in case the file is an index one, since after the file packing the file is re-indexed. For this reason, it is recommended the delete operations to be deferred until before the end of the user session with the system, whenever possible.

4. Programs Performing the Assignment Processing

Assignment processing is the most difficult part of the system. Two programs for each rank perform the assignment process. During this process the necessary criteria (described during the system's design) are examined and a temporary file called ASSIGNED is created. This file contains the assignments of the officers for each rank. Although these programs work in a similar fashion, we did not write only one program to perform the entire assignments, because this program would be very big and difficult to manage. The programs and their function are described below.

a. ASSIGNTS

This program controls the entire assignment processing, and it is called from the MAINPROG program whenever the user selects option 2 from the main-menu. The program calls the ASPRMENU program (assignment processing sub-menu) described earlier, and pauses waiting for the user to make his choice which is stored in the variable 'asprcode'. Then a CASE statement allows the program to branch to the corresponding program and to perform the assignments for the particular rank or exit to the main-menu.

b. ASSIGN01

This program performs the assignments of the officers whose rank is '01' (1st lieutenants). It is called by the ASSIGNTS program in case the user selects option 1 from the assignment processing sub-menu.

First the program calls the MKAUXFL1 program which builds the auxiliary files FILE3, FILE4, and FILE5. The file FILE5 contains all 1st lieutenants whose last assignment took place three or more years before the present date (system's date), their present unit, the requested units for their next assignment, and their assignment weights. The other two auxiliary files (FILE6, FILE7) are copies of the file FILE5. Then it calls the WINDOW1 program which places a frame on the screen into which messages concerning the assignment process are displayed, and starts the assignment processing as follows.

The assignment processing is a two-pass procedure. During the first pass the program loops on the file FILE7 taking one record at a time and tries to satisfy the requests of the officer under examination. That is, it takes the first requested unit and searches the FILE5 file to find an officer

whose present unit is the one requested by the other officer. In case of a successful search, the program searches the FILE6 file to see if the requested unit is also requested by another officer. If it is, then checks the assignment weights. In case he has the greatest weight or his assignment weight is equal to the greatest assignment weight found and his marital status is not equal to 'S' (single), his request is satisfied, his record in the FILE6 file is marked for deletion (so that it will not be examined during the next loop), the record in the FILE5 file is marked for deletion (this means that the requested unit has been granted and it cannot be disposed to another officer), and zero (0) assignment weight is given to the officer. This number is going to be added to the total assignment weight in the corresponding field of the OFFICER file later. This weight is maintained for each officer during his career and it is an important criterion for the assignment process. In case one or more of the conditions described above is false the program takes the second requested unit and the same process is repeated. If the officer's request for the second unit is satisfied, the assignment weight which is given him is one (1) this time. Again if the officer's second request cannot be satisfied, the program takes the third requested unit and the same process is repeated. If his request for the third unit is satisfied, the assignment weight is two (2). Finally, if none of the three requests has been satisfied during the above process, the assignment of the officer is deferred for the second pass. The output of the first pass is the creation of one record per officer into the ASSIGNED file containing among other fields the field 'unitname'. This field, in case of a satisfied request contains the name of the new unit the officer is assigned to, or an asterisk '*' in the case of unresolved assignment.

During the second pass the auxiliary file FILE8 is created which has exactly the same structure with the ASSIGNED file but it contains only the unresolved during the first pass assignments (i.e., UNITNAME = '*'). Then the program loops on this file taking one record at a time and tries for another time to satisfy unsatisfied requests. The process is similar with the first pass. If the requests for an officer cannot be satisfied even this time, the program checks the FILE5 file to find an available unit (not deleted record) in which the officer can serve according to his source and specialty. In the case of a successful search the contents of the 'unitname' field into the ASSIGNED file for the corresponding officer are replaced with the unit name found in FILE5, the record in the FILE5 file is marked for deletion, and the assignment weight three (3) is given to the officer. In case of an unsuccessful search in the FILE5 file, there is no way for the officer to be assigned to a new unit during this year, and the corresponding record into the ASSIGNED file is marked for deletion.

After the last record of the FILE8 file has been examined and the end of file is encountered, the ASSIGNED file is packed, all the auxiliary files are deleted, and control passes to the ASSIGNTS program.

c. MKAUXFL1

This program builds the auxiliary files FILE5, FILE6, and FILE7 which are exactly the same. Those files whose use has been explained above are built by combining the basic database files OFFICER, SERVES, and REQUESTS (copy and join operations). The purpose of those files is to isolate the officers under assignment whose rank is '01' (1st lieutenants), and who have all the requirements for the assignment process, gathering all the data into only three small files. The final

result is that by having all required data into fewer and smaller files we reduce the overhead of the search operations (FIND, LOCATE, and SEEK) and speed up the processing.

The program is called by the ASSIGN01 program. It calls the program WINDOW which displays on the screen a frame into which messages concerning the process of building the auxiliary files are placed. After the auxiliary files have been created the program terminates and control is passed back to the ASSIGN01 program.

d. ASSIGN02

This program performs the assignments of the 2nd lieutenants. It is called by the ASSIGNTS program in case the user selects option 2 from the assignment processing sub-menu.

The process of the program goes in a similar way as the ASSIGN01 program with the following exceptions:

- (1) First, the 2nd lieutenants who graduate from the Military Academy are assigned to the Artillery School for 1-year training. As it is expected no assignment weights are examined, neither requests exist.
- (2) Next, the 2nd lieutenants who complete their 1-year training in the Artillery School are assigned to the Artillery Recruit Training Center. Again no requests or assignment weights are examined.
- (3) Then all 2nd lieutenants whose source is the Military Academy and who have served in the Artillery Recruit Training Center are assigned to the combat units in which they can serve according to the organization table of the units. During this process only the officers' requests are examined, since the assignment weight is zero (0) for all these officers.
- (4) Finally the assignments of the 2nd lieutenants whose source is the Non-Commissioned Officers School and who have completed the necessary time in the same unit (greater than or equal to 3 years) are performed.

e. MKAUXFL2

This program is called by the ASSIGN02 program, and it builds the required auxiliary files for the assignment processing of the 2nd lieutenants. Its function is exactly the same as the MKAUXFL1 described above. The only difference is that the auxiliary files built contain data concerning the 2nd lieutenants. The program calls the WINDOW1 program, and when all the auxiliary files have been built control passes back to the ASSIGN02 program.

f. ASSIGN03

This program performs the assignments of the captains. It is called by the ASSIGNTS program in case the user selects option 3 from the assignment processing sub-menu. Its function is exactly the same as the ASSIGN01 program described earlier. The only difference is that it makes assignments for the captains this time.

The program first calls the MKAUXFL3, and next the WINDOW1 programs. After its termination, control passes to the ASSIGNTS program.

g. MKAUXFL3

This program is called by the ASSIGN03 program during the captains' assignment processing. Its structure and function is exactly the same as that described in MKAUXFL1. The only difference is that the auxiliary files FILE5, FILE6, and FILE7 contain data concerning the captains to be assigned. The program calls the WINDOW program. After its termination, control passes to the ASSIGN03 program.

h. ASSIGN04

This program performs the assignments of the majors and it is called by the ASSIGNTS program in case the user selects option 4 from the assignment processing sub-menu. Its structure and function is almost the same as the ASSIGN01 program with the following exceptions.

- (1) First the majors graduating from the War College are assigned to staff units within the Branch. Assignment weights and officers' requests are taken into consideration.
- (2) Next, the rest of majors whose last assignment took place two or more years before the present date (system's date), are assigned to units inside or outside the Branch, or to the War College for training. Officers assigned to staff units or to units outside the Branch must have graduated from the War College. Assignment weights and officers' requests are examined. An officer is assigned to the War College for training only in case he has been proposed by the Branch. For this reason, the file SELECTED is checked each time a major's assignment is processed.

The program calls the MKAUXFL4, and WINDOW1 programs. When the program terminates, control passes to the ASSIGNTS program.

i. MKAUXFL4

This program is called by the ASSIGN04 program and builds the required auxiliary files for the assignment processing of the majors. Its structure and function is similar to the MKAUXFL1 program. The program calls the WINDOW program. When the program terminates, control is passed to the ASSIGN04 program.

j. ASSIGN05

This program performs the assignments of the lieutenant colonels. It is called by the ASSIGNTS program in case the user selects option 5 from the assignment processing sub-menu. Its structure and function is almost similar to the ASSIGN01 program, with the following exceptions.

- (1) The lieutenant colonels can be assigned to units inside or outside the Branch.
- (2) Officers assigned to staff units or to units outside the Branch must have completed the requirements for their rank, and they must have graduated from the War College.

The program calls the MKAUXFL5, and WINDOW1 programs. After its termination, control passes to the ASSIGNTS program.

k. MKAUXFL5

This program is called by the ASSIGN05 program and builds the required files for the assignment processing of the lieutenant colonels. Its structure and function is similar to the MKAUXFL1 program. The program calls the WINDOW program. When it terminates, control passes to the ASSIGN05 program.

l. ASSIGN06

This program performs the assignments of the colonels. It is called by the ASSIGNTS program in case the user selects option 6 from the assignment processing sub-menu. Its function is similar to the ASSIGN05 program. The program calls the MKAUXFL6, and WINDOW1 programs. When the program terminates, control passes to the ASSIFNTS program.

m. MKAUXFL6

This program is called by the ASSIGN06 program and builds the required auxiliary files for the assignment processing of the colonels. Its structure and function is similar to the MKAUXFL1 program. The program calls the WINDOW program. After its termination, control passes back to the ASSIGN06 program.

5. Programs Producing the Required Lists and Reports

All programs producing the required lists and reports, have been kept small and simple. Using the report generator facility of dBASE III, you can create report formats up to 80 columns wide in a very simple way. You simply give a file name to the required report, and define its format in the way that fits your needs. Defining the format means providing dBASE III with the title of the report, the size of the page, spacing, number of characters per line, what fields of the records will be printed, and what the field headers will be. All the rest of the work is done by dBASE III.

Sample lists and reports produced by the programs described below, are presented in APPENDIX A.

a. REPORTS

This program controls the entire operation of this group of programs. It is called by the MAINPROG (main program) program in case the user selects option 3 from the main-menu. First, the program calls the RGMENU program which is the report generator sub-menu described earlier, and pauses waiting for the user to make his choice which is stored in the variable 'repcod'. Then a CASE statement permits the program

to branch to the appropriate program within the same group of programs according to the users choice or exit to the main-menu. A DO WHILE loop keeps the program running until the user decides to exit to main-menu.

b. LIST1

This program is called by the REPORTS in case the user selects option 1 from the report generator sub-menu, and its purpose is to produce a list of the scheduled assignments for some requested rank. First, the program calls the WINDOW2 program which displays a window on the screen into which program messages and user data or answers are placed. A message prompts the user to specify the rank. Then, according to the rank the user enters, the program builds the auxiliary file AUXFILE1 which contains all necessary data for the requested list. This file is built by combining the OFFICER, REQUESTS, and SERVES file (COPY and JOIN operations). When the file is ready, the message 'PRINTER OUTPUT? (Y/N)' prompts the user to specify if he wants a hardcopy printout. If his answer is 'Y' (yes), another message is displayed requesting from the user to set the printer on and to hit any key to continue. Then the program pauses waiting for the user's action. Finally, the command 'REPORT FORM MKLIST1' causes the requested list to be displayed on the screen, or both displayed on the screen and printed by the printer. MKLIST1 is the format file we have defined in the way we described earlier. After this, the auxiliary file is deleted and control passes to the REPORTS program.

c. LIST2

This program is called by the REPORTS program in case the user selects option 2 from the report generator

sub-menu, and creates a list of officers serving in some requested unit.

The program's function from the user's side of view is similar to the LIST1 program described above. The difference is that this time, the user is prompted to specify the unit name, and that the format file is called MKLIST2. The program calls the WINDOW2 program whose purpose has been explained in the previous program. After the requested output has been produced control passes to the REPORTS program.

d. LIST3

This program provides a list of all Artillery officers in some requested order. It is called by the REPORTS program in case the user selects option 3 from the report generator sub-menu.

First, the program calls the WINDOW2 program. Then a screen of information concerning the possible orders is displayed inside the window, and a message prompts the user to select the order he likes (by seniority, alphabetically, by specialty, etc.). According to the user's choice, a CASE statement allows the program to select the appropriate index file. We do not use the 'SORT' dBASE III command to sort a file in some order, because it is time and space consuming. Instead, we use existing index files which are always presented in the order of the field on which they are indexed. In all other aspects the program works in the same way as the LIST1 and LIST2 programs except that the format file is called MKLIST3 this time.

e. LIST4

This program is called by the REPORTS program in case the user selects option 4 from the report generator

sub-menu. The program produces a list of officers of some requested rank. The process goes exactly in the same way as the LIST1 program. The format file is called MKLIST4.

f. LIST5

This program produces a list of the Artillery battalion commanders, and it is called by the REPORTS program in case the user selects option 5 from the report generator sub-menu. During its execution the user is asked to specify only the device on which he wants the output (screen or printer). The format file is called MKLIST5. In all other aspects the process goes exactly in the same way as in LIST1 program.

g. LIST6

This program is called by the REPORTS program in case the user selects option 6 from the report generator sub-menu. The program provides a list of all officers serving outside the Branch. The report file is called MKLIST6. The user is asked by the program to specify only the device on which he wants the output. The program works in the same way as the LIST5 program.

h. REPORT1

This program creates the service time report for any requested officer. This report contains all the major transactions of an officer occurring during his career. It is called by the REPORTS program in case the user selects option 7 from the report generator sub-menu. The program searches the HISTORIC file and prints out or displays on the screen in an appropriate format all records whose key matches with that

typed by the user when he was asked by the program to enter the officer's serial number.

i. REPORT2

This program is called by the REPORTS program in case the user selects option 8 from the report generator sub-menu. It provides a status report for any requested officer. This report contains all information on an officer's current status. The database files which are searched are the OFFICER and SERVES.

6. Miscellaneous Programs

This group includes all programs which do not perform any processing of data. Their purpose is to provide formatted messages, and display windows on the screen, required during the database processing. These programs are the following:

a. GRFLAG

This program is called by the MAINPROG (main program), and forms on the screen the Hellenic Flag.

b. DELAY

This program is called by most of the programs during the processing, and makes various program messages stay-ing on the screen for a certain time period.

c. DBSTITLE

This program is called by the MAINPROG program and displays on the screen the database system title.

d. MFRAME

This program is called by the INSERTOR, MODIFYOR, and DELETEOR programs, and it displays a frame on the left half of the screen into which messages and formatted records are placed.

e. FRAME

This program is called by the INSERTOR and MODIFYOR programs, and displays a small window into which formatted records from the HISTORIC file are placed.

f. WINDOW

This program is called by the MKAUXFL1, MKAUXFL2, MKAUXFL3, MKAUXFL4, MKAUXFL5, and MKAUXFL6, and its purpose is to display a window on the screen, into which messages informing the user about the processing, are displayed.

g. WINDOW1

This program is called by the ASSIGN01, ASSIGN02, ASSIGN03, ASSIGN04, ASSIGN05, and ASSIGN06 programs during the assignment processing, and it serves the same purpose as the the previous program.

h. WINDOW2

This program is called by the LIST1, LIST2, LIST3, LIST4, LIST5, LIST6, REPORT1, and REPORT2 programs, and it provides the same function as the WINDOW program.

i. WINDOW3

This program is called by the INSERTOR program in case the user's answer to the program's prompt 'DO YOU NEED CODES? (Y/N)' is 'Y'. The program provides a window on the right half of the screen into which all codes concerning rank, source, specialty, and marital status, required during the insertion of new records into the OFFICER file, are displayed.

V. CONCLUSIONS AND RECOMMENDATIONS

This thesis develops a personnel database system model, suitable for implementation within the Artillery Branch Directorate of the Hellenic Army General Staff. This system could also be applied to any Branch Directorate with minor modifications.

The main goal is to increase productivity, effectiveness, accuracy, and speed, as far as personnel management is concerned, as well as to decrease the national expenditure, and release manpower for other purposes. Additionally, the Branch Director will be able to make faster and better informed decisions concerning personnel.

dBASE III was used as the DBMS, since it is a relational model, which is simple and understandable, increases independency, reduces redundancy, and it is very popular in the micro-computer world. In addition, dBASE III contains its own programming language, which is a high level structured language, very efficient for data manipulation.

I have implemented the officers' assignment processing, and the most usually needed lists and reports, but a wide variety of other reports, or simple queries could also be created. Emphasis was given to provide simple and user friendly programs, in order to help the users of the system and make their job easier.

The software life cycle has been taken into account during the program development. Since there was no previous experience on the topic of assignment processing, and no concrete specifications about the organization of the units (the tables used are figurative), I have decided to follow the prototyping approach in order to create the programs. This means that some

of the programs may need further improvements. This can be done in close cooperation with the Branch, which will be the actual user of the system. Programs have been kept small and are easily modified to meet future improvement needs. In this application I have used the top-down design approach which serves the above goal.

In this implementation I have used files whose records include a certain amount of data. Further improvements might add more fields in the records, so that the system can provide expanded information.

This thesis constitutes a good basis for further improvements in the area of personnel management and especially in the field of automation of the officers' assignment processing in the Hellenic Army.

APPENDIX A

DATABASE SYSTEM PROGRAMS

A. MAIN PROGRAM

***** PROGRAM MAINPROG *****

* This is the main program, which controls the operation of
* the entire database system

CLEAR

* Initialize basic dBASE III functions

SET TALK OFF

SET DELIMITER OFF

SET HEADING OFF

SET EXACT ON

* Declare global variables

PUBLIC psw

STORE ' ' TO psw

@ 10,18 SAY 'IKMMMK,'

@ 11,18 SAY 'L9' L9'

@ 12,18 SAY 'HJMMMJ<'

* Check user's authorization

@ 11,30 SAY 'ENTER PASSWORD ->'

SET CONSOLE OFF

ACCEPT TO psw

SET CONSOLE ON

USE userid

LOCATE FOR password = UPPER(psw)

* Unauthorized user. Exit to operating System

IF EOF()

SET COLOR TO W*

@ 11,28 SAY ' UNAUTHORIZED USER . '

DO delay

SET COLOR TO W

QUIT

ENDIF

* Authorized user

STORE .T. TO continue

DO grflag

DO delay

DO dbstitle

```

DO WHILE continue
  DO mmenu
  * perform appropriate function depending on user's choice
  DO CASE
    CASE selection = 1
      DO dbupdate
    CASE selection = 2
      DO assigns
    CASE selection = 3
      DO reports
    CASE selection = 4
      STORE .F. TO continue
    CASE selection = 5
      QUIT
  ENDCASE
ENDDO
SET TALK ON
SET DELIMITER ON
SET EXACT OFF
SET HEADING ON
CLEAR ALL
RETURN

```

B. MAIN-MENU AND SUB-MENU PROGRAMS

1. Main-Menu program

***** PROGRAM MMENU *****

* This program displays the system main-menu on the screen

```
CLEAR
PUBLIC selection
STORE 0 TO selection
@ 4,18 SAY 'IKMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMK,'
@ 5,18 SAY 'L9'
@ 6,18 SAY 'HJMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMT<'
SET COLOR TO W*
@ 5,35 SAY 'MAIN MENU'
SET COLOR TO W
@ 7,18 SAY 'IMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM,'
@ 8,18 SAY ':
@ 9,18 SAY ':      DATABASE UPDATE:.....1      ':
@ 10,18 SAY ':      ASSIGNMENT PROCESSING:.....2      ':
@ 11,18 SAY ':      REPORT GENERATOR:.....3      ':
@ 12,18 SAY ':      END OF DATABASE SESSION:.....4      ':
@ 13,18 SAY ':      EXIT TO DOS:.....5      ':
@ 14,18 SAY ':
@ 15,18 SAY ':
@ 16,18 SAY ':
@ 17,18 SAY 'HMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM<'
SET COLOR TO W+
@ 15,29 SAY 'ENTER YOUR SELECTION ->:',
      GET selection PICTURE '9' RANGE 1,5
      READ
SET COLOR TO W
RETURN
```

2. Sub-Menu Programs

***** PROGRAM UPDTMENU *****

* This ptoqram displays on the screen the update sub-menu

```
CLEAR
PUBLIC updtcode
STORE 0 TO updtcode
@ 4,18 SAY 'IKMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMK,'
@ 5,18 SAY 'L9' L9'
@ 6,18 SAY 'HJMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMJ<'
SET COLOR TO W*
@ 5,35 SAY 'UPDATE MENU'
SET COLOR TO W
@ 7,18 SAY 'IMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM,'
@ 8,18 SAY ':':
@ 9,18 SAY ': INSERT RECORDS INTO OFFICER FILE:.....1 ':
@ 10,18 SAY ': INSERT RECORDS INTO SCHOOLS FILE:.....2 ':
@ 11,18 SAY ': INSERT RECORDS INTO HISTORIC FILE:.....3 ':
@ 12,18 SAY ': MODIFY RECORDS FROM OFFICER FILE:.....4 ':
@ 13,18 SAY ': MODIFY RECORDS FROM REQUESTS FILE:.....5 ':
@ 14,18 SAY ': DELETE RECORDS FROM OFFICER FILE:.....6 ':
@ 15,18 SAY ': EXIT TO MAINMENU:.....7 ':
@ 16,18 SAY ':':
@ 17,18 SAY ':':
@ 18,18 SAY ':':
@ 19,18 SAY 'HMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM<'
SET COLOR TO W+
@ 17,29 SAY 'ENTER YOUR SELECTION ->:',
      GET updtcode PICTURE '9' RANGE 1,7
      READ
SET COLOR TO W
RETURN
```

***** PROGRAM ASPRMENU *****

* This program displays on the screen the assignment processing sub-menu
*

```
CLEAR
PUBLIC asprcode
STORE 0 TO asprcode
@ 4,18 SAY 'IKMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMK,'
@ 5,18 SAY 'L9'
@ 6,18 SAY 'HJMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMJ<'
SET COLOR TO W*
@ 5,28 SAY 'ASSIGNMENT PROCESSING MENU'
SET COLOR TO W
@ 7,18 SAY 'IMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM,'
@ 8,18 SAY ':':
@ 9,18 SAY ': 1st LIEUTENANT ASSIGNMENT PROCESSING:...1 ':
@ 10,18 SAY ': 2nd LIEUTENANT ASSIGNMENT PROCESSING:...2 ':
@ 11,18 SAY ': CAPTAIN ASSIGNMENT PROCESSING:.....3 ':
@ 12,18 SAY ': MAJOR ASSIGNMENT PROCESSING:.....4 ':
@ 13,18 SAY ': LIEUT. COLONEL ASSIGNMENT PROCESSING:...5 ':
@ 14,18 SAY ': COLONEL ASSIGNMENT PROCESSING:.....6 ':
@ 15,18 SAY ': EXIT TO MAIN MENU:.....7 ':
@ 16,18 SAY ':':
@ 17,18 SAY ':':
@ 18,18 SAY ':':
@ 19,18 SAY 'HMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM<'
SET COLOR TO W+
@ 17,29 SAY 'ENTER YOUR SELECTION ->',
      GET asprcode PICTURE '9' RANGE 1,7
      READ
SET COLOR TO W
RETURN
```

***** PROGRAM RGMENU *****

* This program displays on the screen the report generator
* sub-menu

```
CLEAR
PUBLIC repcode
STORE 0 TO repcode
@ 3,14 SAY 'IKMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMK,'
@ 4,14 SAY 'L9'
@ 5,14 SAY 'HJMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMJ<'
SET COLOR TO W*
@ 4,30 SAY 'REPORT GENERATOR MENU'
SET COLOR TO W
@ 6,14 SAY 'IMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM,'
@ 7,14 SAY ':'
@ 8,14 SAY ':LIST OF SCHEDULED ASSIGNMENTS:.....1:'
@ 9,14 SAY ':LIST OF OFFICERS OF A SPECIFIC UNIT:.....2:'
@ 10,14 SAY ':LIST OF OFFICERS IN ANY DESIRED ORDER:.....3:'
@ 11,14 SAY ':LIST OF OFFICERS OF A SPECIFIC RANK:.....4:'
@ 12,14 SAY ':LIST OF BATTALION COMMANDERS:.....5:'
@ 13,14 SAY ':LIST OF OFFICERS SERVING OUTSIDE THE BRANCH:.6:'
@ 14,14 SAY ':OFFICER'S SERVICE TIME REPORT:.....7:'
@ 15,14 SAY ':OFFICER'S STATUS REPORT:.....8:'
@ 16,14 SAY ':EXIT TO MAIN MENU:.....9:'
@ 17,14 SAY ':'
@ 18,14 SAY ':'
@ 19,14 SAY ':'
@ 20,14 SAY 'HMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM<'
SET COLOR TO W+
@ 18,27 SAY 'ENTER YOUR SELECTION ->',
    GET repcode PICTURE '9' RANGE 1,9
    READ
SET COLOR TO W
RETURN
```

C. PROGRAMS SUPPORTING THE UPDATE OPERATIONS

***** PROGRAM DBUPDATE *****

* This program controls the system's update operations

```
CLEAR
STORE ' ' TO updtcont
PUBLIC updtcode
DO WHILE updtcont # 'n'
    DO updtmenu
    DO CASE
        CASE updtcode = 1
            DO insertor
        CASE updtcode = 2
            DO insertsr
        CASE updtcode = 3
            DO inserthr
        CASE updtcode = 4
            DO modifyor
        CASE updtcode = 5
            DO modifyrr
        CASE updtcode = 6
            DO deleteor
        CASE updtcode = 7
            STORE 'n' TO updtcont
    ENDCASE
ENDDO
RETURN
```


***** PROGRAM INSERTOR *****

* This program adds records to OFFICER file, updates all other
 * affected database files, as well as records log-data into
 * USERLOG file

CLEAR

STORE 'y' TO insertcont

* open required for the processing database files

SELECT 1

USE historic INDEX historic

SELECT 2

USE requests INDEX requests

SELECT 3

USE officer INDEX officer

SELECT 4

USE userid

SELECT 5

USE userlog

SELECT 6

USE serves INDEX serves

STORE .F. TO done

DO WHILE UPPER(insertcont) = 'Y'

SELECT 3

DO mframe

@ 3,5 SAY 'INSERT NEW RECORD TO OFFICER FILE'

@ 4,5 SAY 'MM'

* Initialize memory variables

STORE ' ' TO tunit

STORE ' ' TO torder

STORE ' ' TO tname

STORE ' ' TO tserno

STORE ' ' TO trunk

STORE ' ' TO tnomyear

STORE ' ' TO tspec

STORE ' ' TO tsource

STORE ' ' TO ndate

STORE ' ' TO pdate

STORE ' ' TO tcity

STORE ' ' TO tcounty

STORE ' ' TO tmarst

STORE 0 TO tchild

STORE .F. TO twwife

@ 20,7 SAY 'ENTER SERIAL NUMBER ->' GET tserno

PICTURE '99999'

READ

```

* check whether record exists into OFFICER file
FIND &tserno
IF EOF()
    * record does not exist
    @ 20,7 SAY '
    STORE ' ' TO answer
    @ 12,45 SAY 'DO YOU NEED CODES? (Y/N) ==>' GET answer
    READ
    IF UPPER(answer) = 'Y'
        DO window3
    ELSE
        @ 12,45 CLEAR
    ENDIF
    * read user's data form the keyboard
    @ 5,5 SAY 'name : ' GET tname
    PICTURE 'AAAAAAAAAAAAAAAAAAAA'
    @ 6,5 SAY 'serial # : ' GET tserno PICTURE '99999'
    @ 7,5 SAY 'rank : ' GET trank PICTURE '99'
    @ 8,5 SAY 'nomination year: ' GET tnomyear PICTURE '99'
    @ 9,5 SAY 'specialty : ' GET tspec PICTURE 'A'
    @ 10,5 SAY 'source : ' GET tsource PICTURE 'AAAA'
    @ 11,5 SAY 'nomination date: ' GET ndate
    PICTURE '99/99/99'
    @ 12,5 SAY 'promotion date : ' GET ndate
    PICTURE '99/99/99'
    @ 13,5 SAY 'origin (city) : ' GET tcity PICTURE 'XXXXXX'
    @ 14,5 SAY 'origin (county): ' GET tcounty
    PICTURE 'XXXXXXXX'
    @ 15,5 SAY 'marital status : ' GET tmarst PICTURE 'A'
    @ 16,5 SAY '# of children : ' GET tchild PICTURE '9'
    @ 17,5 SAY 'working wife? : ' GET twwife
    READ

    * append new record to OFFICER file
    APPEND BLANK
    REPLACE name WITH tname, serno WITH tserno
    REPLACE rank WITH trank, nomyear WITH tnomyear
    REPLACE specialty WITH tspec, source WITH tsource
    REPLACE asnweight WITH 0 nomdate WITH CTOD(ndate)
    REPLACE promdate WITH CTOD(ndate), origcity WITH tcity
    REPLACE origcounty WITH tcounty, marstat WITH tmarst
    REPLACE children WITH tchild, workwife WITH twwife
    STORE .T. TO done

    * create an etry into REQUESTS file
    SELECT 2
    FIND &tserno
    IF EOF()
        APPEND BLANK
        REPLACE serno WITH tserno
    ENDIF

```

AD-A174 137

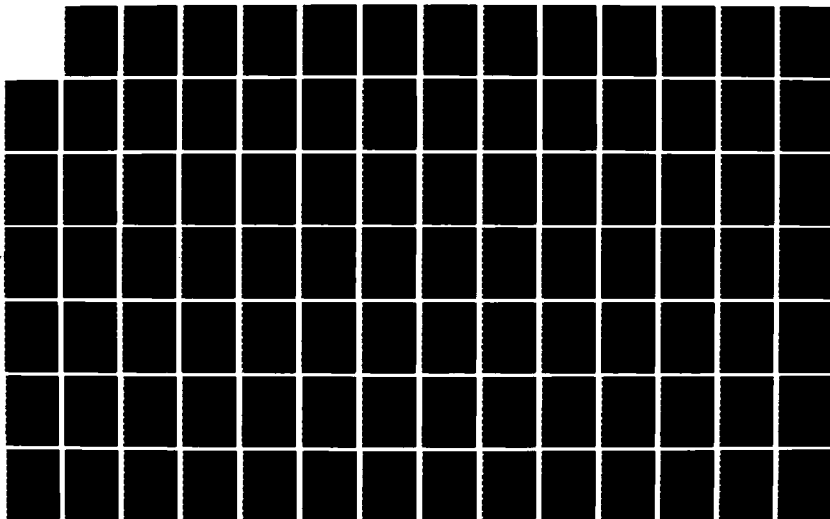
IMPLEMENTATION OF A PERSONNEL DATABASE SYSTEM
PERFORMING THE ANNUAL REASS (U) NAVAL POSTGRADUATE
SCHOOL MONTEREY CA I G STROUZAS 28 JUN 86

2/3

UNCLASSIFIED

F/G 5/9

NL





XEROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

```

* add record to SERVES file if officer's source is NCOS
IF tsource = 'NCOS'
  SELECT 6
  FIND &tserno
  IF EOF()
    STORE ' ' TO tunit
    @ 20,7 SAY 'ENTER UNIT NAME -> ' GET tunit
    READ
    APPEND BLANK
    REPLACE serno WITH tserno,unitname WITH tunit,,
      enroldate WITH CTOD(ndate)
  ENDIF
ENDIF

* add record to HISTORIC file (transact. is NOMINATION)
SELECT 1
STORE 'NOMINATION ' TO trans
SEEK tserno + trans + ndate
IF EOF()
  STORE ' ' TO torder
  @ 20,5 SAY 'ENTER ORDER->' GET torder
  READ
  @ 20,5 SAY '
  APPEND BLANK
  REPLACE serno WITH tserno, rank WITH trank,,
    transtype WITH trans,transdate WITH CTOD(ndate),,
    orderid WITH torder,unit WITH tunit
  @ 2,45 CLEAR

  * display new record of HISTORIC file
  DO frame
  @ 11,45 SAY 'serial #:' GET serno
  @ 12,45 SAY 'rank      :' GET rank
  @ 13,45 SAY 'transac.:' GET transtype
  @ 14,45 SAY 'unit      :' GET unit
  @ 15,45 SAY 'date       :' GET transdate
  @ 16,45 SAY 'order      :' GET orderid
ELSE
  ? '*** ERROR: RECORD EXISTS IN HISTORIC FILE'
  DISPLAY
  DO delay
  @ 22,0 CLEAR
ENDIF

* record exists into OFFICER file and cannot be added again
ELSE
  @ 20,6 SAY 'RECORD ALREADY EXISTS
  DO delay
ENDIF

```

```

@ 20,6 SAY '      MORE INSERTIONS? (Y/N) ->'
SET CONSOLE OFF
WAIT TO insertcont
SET CONSOLE ON
CLEAR
ENDDO
* record log-data into the USERLOG file
IF done
  SELECT 4
  LOCATE FOR password = UPPER(psw)
  SELECT 5
  APPEND BLANK
  REPLACE username WITH D->username, task WITH 'INSERTION',,
    progame WITH 'INSERTOR',logdate WITH DATE(),,
    logtime WITH TIME()
ENDIF
CLOSE DATABASES
RETURN

```

***** PROGRAM INSERTSR *****

CLEAR

```
SELECT 1
USE userid
SELECT 2
USE userlog
SELECT 3
USE officer INDEX officer
SELECT 4
USE schools
```

[illegible]

STORE .F. TO done

Q 5.24 SAY 'INSERT NEW RECORD TO SCHOOLS FILE'

```
STORE ' ' TO tserno
STORE ' ' TO tschname
STORE ' ' TO tdegree
STORE ' ' TO tobject
STORE ' ' TO tcountry
STORE 0 TO tduration
STORE ' ' TO tqdate
```

READ

```

* check if record exists in OFFICER file
SELECT 3
FIND &tserno
IF .NOT. EOF()
    * record exists in OFFICER file
    * check if the record to be added exists in SCHOOLS file
    @ 16,26 SAY ' ENTER SCHOOL NAME ->' GET tschname
    READ
    SELECT 4
    LOCATE FOR serno = tserno .AND. schoolname = tschname
    IF EOF()
        * record does not exist in SCHOOLS file
        * read user's data from the keyboard
        @ 16,26 SAY '
        @ 7,23 SAY 'serial #           :' GET tserno
        @ 8,23 SAY 'school-name        :' GET tschname
        @ 9,23 SAY 'degree             :' GET tdegree
        @ 10,23 SAY 'object of studies:' GET tobject
        @ 11,23 SAY 'country           :' GET tcountry
        @ 12,23 SAY 'duration          :' GET tduration
                                PICTURE '99'
        @ 13,23 SAY 'graduation date  :' GET tgdate
                                PICTURE '99/99/99'

        READ

        * append new record to SCHOOLS file
        APPEND BLANK
        REPLACE serno WITH tserno, schoolname WITH tschname,
            degree WITH tdegree, object WITH tobject,,
            country WITH tcountry, duration WITH tduration,,
            graddate WITH CTOD(tgdate)
        STORE .T. TO done

    * record exist in SCHOOLS file
    ELSE
        @ 16,26 SAY 'RECORD ALREADY EXISTS
        DO delay
    ENDIF

* record does not exist in OFFICER file, and we cannot
* add new recor to SCHOOLS file
ELSE
    @ 16,21 SAY 'RECORD DOES NOT EXIST IN OFFICER FILE'
    DO delay
    @ 16,21 SAY '
ENDIF
@ 16,26 SAY 'MORE INSERTIONS? (Y/N) ->'
SET CONSOLE OFF
WAIT TO insertcont
SET CONSOLE ON
ENDDO

```



```
* update USERLOG file
IF done
  SELECT 1
  LOCATE FOR password = UPPER(psw)
  SELECT 2
  APPEND BLANK
  REPLACE username WITH B->username, task WITH 'INSERTION',,
    progame WITH 'INSERTSR',logdate WITH DATE(),,
    logtime WITH TIME()
ENDIF
CLOSE DATABASES
RETURN
```

***** PROGRAM INSERTHR *****

* This program adds new records to HISTORIC file, and updates
* USERLOG file

CLEAR

* open required for the processing files

SELECT 1

USE serves INDEX serves

SELECT 2

USE historic INDEX historic

SELECT 3

USE officer INDEX officer

SELECT 4

USE userid

SELECT 5

USE userlog

STORE 'y' TO insertcont

@ 2,17 SAY 'THE ONLY LEGAL TRANSACTIONS FOR WHICH A RECORD'

@ 3,17 SAY ' CAN BE ADDED FROM THIS PROGRAM ARE:'

SET COLOR TO W+

@ 4,17 SAY ' . RETIREMENT'

@ 5,17 SAY ' . DEATH'

SET COLOR TO W

STORE .F. TO done

SELECT 2

DO WHILE UPPER(insertcont) = 'Y'

* display window on the screen

@ 7,18 SAY 'IMM,'

@ 8,18 SAY ': INSERT RECORDS INTO HISTORIC FILE :'

@ 9,18 SAY ': MM :'

@ 10,18 SAY ': :'

@ 11,18 SAY ': :'

@ 12,18 SAY ': :'

@ 13,18 SAY ': :'

@ 14,18 SAY ': :'

@ 15,18 SAY ': :'

@ 16,18 SAY ': :'

@ 17,18 SAY ': :'

@ 18,18 SAY 'HMMM<'

* initialize memory variables

STORE .F. TO duplicate

STORE ' ' TO tserno

STORE ' ' TO trank

STORE ' ' TO trans

STORE ' ' TO tdate

STORE ' ' TO torder

@ 17,20 SAY 'ENTER SERIAL NUMBER ->' GET tserno

PICTURE '99999'

READ

```

* check if record exists in OFFICER file
SELECT 3
FIND &tserno
IF .NOT. EOF()
    * record exists in OFFICER file
    SELECT 1
    FIND &tserno
    SELECT 2
    @ 17,20 SAY ' ENTER TRANSACTION ->' GET trans
    READ

    * check whether transaction is legal or not
    IF trans = 'RETIREMENT' .OR. trans = 'DEATH'
        * legal transaction
        @ 17,20 SAY '
        @ 17,20 SAY ' ENTER DATE ->' GET tdate
        PICTURE '99/99/99'

        READ
        @ 17,20 SAY '

        * check for duplicate record in HISTORIC file
        SEEK tserno + trans + tdate
        IF EOF()
            * no duplicate record
            * red user's data from keyboard
            @ 11,20 SAY 'rank : ' GET trank PICTURE '99'
            @ 12,20 SAY 'transaction : ' GET trans
            PICTURE 'AAAAAAAAAAAA'
            @ 14,20 SAY 'date : ' GET tdate
            PICTURE '99/99/99'
            @ 15,20 SAY 'order : ' GET torder
            READ
            * append new record to HISTORIC file
            APPEND BLANK
            REPLACE serno WITH tserno, rank WITH trank,,
            transtype WITH trans, unit WITH A->unitname,,
            transdate WITH CTOD(tdate),orderid WITH torder
            @ 10,20 SAY 'serial # : ' GET serno
            @ 13,20 SAY 'unit : ' GET unit
            STORE .T. TO done

            * record to be added already exists
        ELSE
            @ 17,20 SAY '*** ERROR: DUPLICATE RECORD
            DO delay
        ENDIF
    * illegal transaction
    ELSE
        @ 17,20 SAY 'ILLEGAL TRANSACTION
        DO delay
    ENDIF

```

```

* officer does not exist in OFFICER file
ELSE
  @ 17,20 SAY 'RECORD DOES NOT EXIST IN OFFICER FILE'
  DO delay
  @ 17,20 SAY '
ENDIF
@ 17,20 SAY 'MORE INSERTIONS? (Y/N)'
SET CONSOLE OFF
WAIT TO insertcont
SET CONSOLE ON
CLEAR
ENDDO

* update USERLOG file
IF done
  SELECT 4
  LOCATE FOR password = UPPER(psw)
  SELECT 5
  APPEND BLANK
  REPLACE username WITH D->username, task WITH 'INSERTION',,
    progame WITH 'INSERTHR', logdate WITH DATE(),,
    logtime WITH TIME()
ENDIF
CLOSE DATABASES
RETURN

```

***** PROGRAM MODIFYOR *****

* This program modifies records in the OFFICER file, adds new
 * records to HISTORIC file in case of promotion, and updates
 * USERLOG file

CLEAR

* open required for the processing files

SELECT 1

USE serves INDEX serves

SELECT 2

USE historic INDEX historic

SELECT 3

USE officer index officer

SELECT 4

USE userid

SELECT 5

USE userlog

STORE .F. TO done

STORE 'y' TO modicont

DO WHILE UPPER(modicont) = 'Y'

SELECT 3

DO mframe

@ 3,6 SAY ' MODIFY RECORD IN OFFICER FILE'

@ 4,6 SAY ' MMM'

STORE ' ' TO tserno

@ 20,7 SAY 'ENTER SERIAL NUMBER ->' GET tserno

READ

* check if record to be modified exists in OFFICER file

FIND &tserno

IF .NOT. EOF()

* record exists in OFFICER file

* initialize memory variables

STORE 0 TO attrib

STORE ' ' TO trans

STORE ' ' TO tunit

STORE ' ' TO torder

STORE name TO tname

STORE rank TO trunk

STORE promdate TO tdate

STORE marstat TO tmarst

STORE children TO child

STORE workwife TO twwife

* display which attributes can be modified

@ 20,7 SAY '

@ 5,8 SAY ' MODIFIABLE ATTRIBUTES'

@ 6,8 SAY ' -----'

@ 7,8 SAY 'name 1'

@ 8,8 SAY 'rank and promotion date 2'

@ 9,8 SAY 'marital and family status 3'

```

* read user' s selection from keyboard
@ 20,7 SAY '   ENTER YOUR SELECTION ->' GET attrib,
  PICTURE '9' RANGE 1,3
READ

* clear displayed attributes
@ 5,8 SAY '
@ 6,8 SAY '
@ 7,8 SAY '
@ 8,8 SAY '
@ 9,8 SAY '
@ 20,7 SAY '

DO CASE
  CASE attrib = 1
    * replace old name with the new one
    @ 5,5 SAY 'name           :' GET tname
    READ
    REPLACE name WITH tname
  CASE attrib = 2
    * replace old rank and promotin date
    @ 7,5 SAY 'rank           :' GET trank,
      PICTURE '99'
    @ 12,5 SAY 'promotion date :' GET tdate,
      PICTURE '99/99/99'
    READ
    IF rank # trank
      STORE .T. TO addhist
    ENDIF
    REPLACE rank WITH trank, promdate WITH tdate
  CASE attrib = 3
    * replace family data with the new ones
    @ 16,5 SAY 'marital status :' GET tmarst
    @ 17,5 SAY '# of children  :' GET child
      PICTURE '9'
    @ 18,5 SAY 'working wwife  :' GET twwife
    READ
    REPLACE marstat WITH tmarst, children WITH child,
      workwife WITH twwife
ENDCASE
STORE .T. TO done

* add new record to HISTORIC file in case of promotion
IF attrib = 2
  STORE 'PROMOTION ' TO trans
  SELECT 1
  FIND &tserno
  STORE unitname TO tunit

```

```

* check for duplicate transaction
SELECT 2
SEEK tserno + trans + DTOC(tdate)
IF EOF()
    * no duplicate transaction
    STORE ' ' TO torder
    @ 20,5 SAY 'ENTER ORDER->' GET torder
    READ
    @ 20,5 SAY '
    APPEND BLANK
    REPLACE serno WITH tserno, rank WITH trank,
            transtype WITH trans, unit WITH tunit,
            transdate WITH tdate, orderid WITH torder

    * display new record added to HISTORIC file
    DO frame
    @ 11,45 SAY 'serial #:' GET serno
    @ 12,45 SAY 'rank      :' GET rank
    @ 13,45 SAY 'transact:' GET transtype
    @ 14,45 SAY 'unit      :' GET unit
    @ 15,45 SAY 'date      :' GET transdate
    @ 16,45 SAY 'order     :' GET orderid

    * duplicate transaction
    ELSE
        @ 20,7 SAY '*** ERROR: TRANSACTION EXISTS'
        DO delay
    ENDIF
ENDIF

* display modified OFFICER file record
SELECT 3
@ 3,6 SAY '          UPDATED RECORD
@ 5,5 SAY 'name          :' GET name
@ 6,5 SAY 'serial #      :' GET serno
@ 7,5 SAY 'rank          :' GET rank
@ 8,5 SAY 'nomination year:' GET nomyear
@ 9,5 SAY 'specialty      :' GET specialty
@ 10,5 SAY 'source        :' GET source
@ 11,5 SAY 'nomination date:' GET nomdate
@ 12,5 SAY 'promotion date :' GET promdate
@ 13,5 SAY 'assign. weight :' GET asnweight
@ 14,5 SAY 'origin (city)  :' GET origcity
@ 15,5 SAY 'origin (county):' GET origcounty
@ 16,5 SAY 'marital status :' GET marstat
@ 17,5 SAY '# of children  :' GET children
@ 18,5 SAY 'working wife?  :' GET workwife

```

```

* record to be modified does not exist in OFFICER file
ELSE
  @ 20,5 SAY 'RECORD DOES NOT EXIST IN DATABASE'
  DO delay
ENDIF

@ 20,5 SAY '
@ 20,7 SAY 'MORE MODIFICATIONS? (Y/N) ->'
SET CONSOLE OFF
WAIT TO modicont
SET CONSOLE ON
CLEAR
ENDDO

* update USERLOG file
IF done
  SELECT 4
  LOCATE FOR password = UPPER(psw)
  SELECT 5
  APPEND BLANK
  REPLACE username WITH D->username,,
    task WITH 'MODIFICATION',,
    progame WITH 'MODIFYOR', logdate WITH DATE(),,
    logtime WITH TIME()
ENDIF
CLOSE DATABASES
RETURN

```



```

* read user's data from keyboard
@ 7,26 SAY 'serial #          :' GET tserno
@ 8,26 SAY 'submission date    :' GET tdate
                                PICTURE '99/99/99'
@ 9,26 SAY '1st requested unit :' GET tunit1
                                PICTURE 'XXXXXXXX'
@ 10,26 SAY '2nd requested unit :' GET tunit2
                                PICTURE 'XXXXXXXX'
@ 11,26 SAY '3rd requested unit :' GET tunit3
                                PICTURE 'XXXXXXXX'
READ

* replace changed fields
REPLACE submdate WITH tdate, unit1 WITH tunit1,,
        unit2 WITH tunit2, unit3 WITH tunit3
STORE .T. TO done

* record to be modified does not exist in REQUESTS file
ELSE
    @ 14,24 SAY 'RECORD DOES NOT EXIST IN DATABASE'
    DO delay
ENDIF
@ 14,24 SAY '
@ 14,26 SAY 'MORE MODIFICATIONS? (Y/N) ->'
SET CONSOLE OFF
WAIT TO modicont
SET CONSOLE ON
CLEAR
ENDDO

* update USERLOG file
IF done
    SELECT 2
    LOCATE FOR password = UPPER(psw)
    SELECT 3
    APPEND BLANK
    REPLACE username WITH B->username,,
            task WITH 'MODIFICATION',,
            progame WITH 'MODIFYRR',logdate WITH DATE(),,
            logtime WITH TIME()
ENDIF
CLOSE DATABASES
RETURN

```

***** PROGRAM DELETEOR *****

* This program deletes recods from OFFICER file, and from all
 * other database files whose field 'serno' matches with the
 * key of the deleted from OFFICERS file record. It also updates
 * USERLOG file

CLEAR

* open required for the processing files

SELECT 1

USE serves INDEX serves

SELECT 2

USE requests INDEX requests

SELECT 3

USE schools

SELECT 4

USE historic INDEX historic

SELECT 5

USE officer INDEX officer

STORE 'y' TO deletecont

STORE 0 TO cnt1, cnt2, cnt3, cnt5

STORE .F. TO done

DO WHILE UPPER(deletecont) = 'Y'

DO mframe

@ 3,7 SAY 'DELETE RECORD FROM OFFICER FILE'

@ 4,7 SAY 'MM'

STORE ' ' TO tserno

@ 20,7 SAY ' ENTER SERIAL NUMBER ->' GET tserno
 PICTURE '99999'

* check if record to be deleted exists in OFFICER file

READ

FIND &tserno

IF .NOT. EOF()

* record exists in OFFICER file

* display record to be deleted

@ 20,7 SAY '

@ 5,5 SAY 'name : ' GET name

@ 6,5 SAY 'serial # : ' GET serno

@ 7,5 SAY 'rank : ' GET rank

@ 8,5 SAY 'nomination year: ' GET nomyear

@ 9,5 SAY 'specialty : ' GET specialty

@ 10,5 SAY 'source : ' GET source

@ 11,5 SAY 'nomination date: ' GET nomdate

@ 12,5 SAY 'promotion date : ' GET promdate

@ 13,5 SAY 'origin (city) : ' GET origcity

@ 14,5 SAY 'origin (county): ' GET origcounty

@ 15,5 SAY 'marital status : ' GET marstat

```

* be sure that user wants the record to be deleted
@ 20,7 say '          DELETE? (Y/N) ->'
SET CONSOLE OFF
WAIT TO confirm
SET CONSOLE ON

* delete record from OFFICER file
IF UPPER(confirm) = 'Y'
  DELETE
  STORE .T. TO done
  STORE cnt5 + 1 TO cnt5

  * delete record from SERVES file
  SELECT 1
  FIND &tserno
  IF .NOT. EOF()
    DELETE
    STORE cnt1 + 1 TO cnt1
  ENDIF

  * delete record from REQUESTS file
  SELECT 2
  FIND &tserno
  IF .NOT. EOF()
    DELETE
    STORE cnt2 + 1 TO cnt2
  ENDIF

  * delete records from SCHOOLS file
  SELECT 3
  LOCATE FOR serno = tserno
  IF .NOT. EOF()
    DELETE ALL FOR serno = tserno
    STORE cnt3 + 1 TO cnt3
  ENDIF

  * delete records from HISTORIC file
  SELECT 4
  DELETE ALL FOR serno = tserno
ENDIF

* record to be deleted does not exist in OFFICER file
ELSE
  @ 20,6 SAY 'RECORD DOES NOT EXIST IN DATABASE'
  DO delay
  @ 20,6 SAY '
ENDIF

```

```

@ 20,7 SAY '    MORE DELETIONS? (Y/N) ->'
SET CONSOLE OFF
WAIT TO deletecont
SET CONSOLE ON
CLEAR
SELECT 5
ENDDO

* pack database files which records have been deleted from
IF cnt5 # 0
  @ 10,18 SAY 'DATABASE FILES ARE BEING PACKED AND REINDEXED'
  @ 11,18 SAY '    BE PATIENT, IT WILL TAKE SOME TIME'
  PACK
  SELECT 1
  IF cnt1 # 0
    PACK
  ENDIF
  SELECT 2
  IF cnt2 # 0
    PACK
  ENDIF
  SELECT 3
  IF cnt3 # 0
    PACK
  ENDIF
  SELECT 4
  PACK
ENDIF
CLOSE DATABASES

* update USERLOG file
IF done
  SELECT 1
  USE userid
  LOCATE FOR password = UPPER(psw)
  SELECT 2
  USE userlog
  APPEND BLANK
  REPLACE username WITH A->username, task WITH 'DELETION',,
    progname WITH 'DELETEOR', logdate WITH DATE(),,
    logtime WITH TIME()
ENDIF
CLOSE DATABASES
RETURN

```

D. PROGRAMS PERFORMING THE ASSIGNMENT PROCESSING

***** PROGRAM ASSIGNTS *****

* This program controls the assignment processing operations

CLEAR

PUBLIC asprcode

DO asprmenu

DO WHILE asprcode # 7.

DO CASE

CASE asprcode = 1

DO assign01

CASE asprcode = 2

DO assign02

CASE asprcode = 3

DO assign03

CASE asprcode = 4

DO assign04

CASE asprcode = 5

DO assign05

CASE asprcode = 6

DO assign06

ENDCASE

DO asprmenu

ENDDO

RETURN

***** PROGRAM ASSIGN01 *****

* This program performs the assignments of the 1st lieutenants
* and updates the USERLOG file

* build required temporary files
DO mkauxfl2

* display a window in the screen
DO window1

* assign 1st lieutenants to units in which they can serve by
* combining requests and assignment weights
@ 5,23 SAY 'ASSIGNMENTS FOR THE 1st LIEUTENANTS'
@ 6,23 SAY ' ARE BEING PROCESSED'

* open required for the processing files
SELECT 1
USE assigned INDEX assigned
SELECT 2
USE file5
SELECT 3
USE file6
SELECT 4
USE file7
STORE .T. TO ok
@ 8,23 SAY '
@ 8,23 SAY ' FIRST PASS'
STORE '08/15/86' TO tdate

* get each officer under assignment and determine the unit to
* assigned to or defer it for the second pass

DO WHILE .NOT. EOF()
* initialize memory variables
STORE .F. TO granted
STORE .T. TO preference
STORE serno TO tserno
STORE rank TO trunk
STORE source TO tsource
STORE specialty TO tspec
STORE asnweight TO weight
STORE marstat TO marst
STORE ' ' TO tunit,tu1,tu2,tu3
STORE unit1 TO tu1
STORE unit2 TO tu2
STORE unit3 TO tu3
STORE .F. TO resolved
STORE 1 TO loopcnt

```

* determine which of the requested units will be examined
*           next (1st, 2nd, or 3rd)
DO WHILE .NOT. resolved .AND. loopcnt < 4
  IF loopcnt = 1
    STORE tu1 TO tu
  ELSE
    IF loopcnt = 2
      STORE tu2 TO tu
    ELSE
      STORE tu3 TO tu
    ENDIF
  ENDIF
ENDIF

* check if requested by the officer unit is available,
* i.e., examine if there exists an officer under
* assignment whose present unit is the requested unit
SELECT 3
LOCATE FOR specialty=tspec .AND. source=tsource .AND.,
  unitname=tu .AND. serno#tserno .AND. .NOT. DELETED()
IF .NOT. EOF()
  STORE serno TO idno
  SELECT 2
  LOCATE FOR serno = idno .AND. .NOT. DELETED()

  IF .NOT. EOF()
    * someone else from the requested unit is to be moved
    * check if someone else has requested the same unit
    SELECT 3
    STORE .F. TO done
    DO WHILE .NOT. EOF() .AND. .NOT. done
      IF .NOT. DELETED()
        IF specialty = tspec .AND. serno # tserno,
          .AND. source = tsource
          IF unit1=tu .OR. unit2=tu .OR. unit3=tu
            STORE .T. TO done
          ENDIF
        ENDIF
      ENDIF
    ENDWHILE
    SKIP
  ENDDO

  IF .NOT. done
    * nobody else requests the same unit, requested
    * unit is granted, record is marked for deletion
    STORE tu TO tunit
    DO CASE
      CASE loopcnt = 1
        STORE 0 TO tweight
      CASE loopcnt = 2
        STORE 1 TO tweight
    ENDCASE
  ENDIF
ENDIF

```



```

        CASE loopcnt = 3
            STORE 2 TO tweight
        ENDCASE
        STORE .T. TO granted
        LOCATE FOR serno = tserno
        DELETE
        SELECT 2
        LOCATE FOR serno = idno
        DELETE
        STORE .T. TO resolved
    ELSE
        * somebody else requests the same unit, check
        * assignment weights to see whom will be given
        * the preference to
        DO WHILE ((.NOT. EOF()) .AND. (preference))
            IF .NOT. DELETED()
                IF serno # tserno .AND. specialty=tspec,
                    .AND. source = tsource
                    IF unit1=tu.OR.unit2=tu.OR.unit3=tu
                        IF asnweight > weight
                            STORE .F. TO preference
                        ELSE
                            IF asnweight = weight
                                IF marst = 'S' .AND.,
                                    marstat # 'S'
                                    STORE .F. TO preference
                                ENDIF
                            ENDIF
                        ENDIF
                    ENDIF
                ENDIF
            ENDIF
        SKIP
    ENDDO
    IF preference
        * officer requested this unit has the preferen-
        * ce, unit is granted, record is marked for
        * deletion
        STORE tu TO tunit
        DO CASE
            CASE loopcnt = 1
                STORE 0 TO tweight
            CASE loopcnt = 2
                STORE 1 TO tweight
            CASE loopcnt = 3
                STORE 2 TO tweight
        ENDCASE
        STORE .T. TO granted
        LOCATE FOR serno = tserno
        DELETE

```

```

        SELECT 2
        LOCATE FOR serno = idno
        DELETE
        STORE .T. TO resolved
    ENDIF
ENDIF
ENDIF
STORE loopcnt + 1 TO loopcnt
ENDDO

IF .NOT. resolved
* unresolved assignment, defer it for the next pass
    STORE '*' TO tunit
    STORE 3 TO tweight
    STORE .F. TO ok
ENDIF

* build record in the file containing the assignments
SELECT 1
APPEND BLANK
REPLACE serno WITH tserno, rank WITH trunk,,
        source WITH tsource, specialty WITH tspec
REPLACE unitname WITH tunit, asnweight WITH tweight,,
        asgndate WITH CTOD(tdate)
SELECT 4
SKIP
ENDDO

* second pass

IF .NOT. ok
* unresolved assignments exist
    @ 8,23 SAY '                SECOND PASS'
    STORE .F. TO resolved

    SELECT 1
    * build temporary file containing unresolved assignments
    COPY TO file8 FOR unitname = '*' .AND. rank = '02'
    SELECT 5
    USE file8
    STORE .F. TO deletion

    * get one record at a time and try to satisfy the request
    DO WHILE .NOT. EOF()
        * initialize memory variables
        STORE ' ' TO tsource
        STORE serno TO tserno
        STORE specialty TO tspec
        STORE source TO tsource
        STORE ' ' TO tunit

```

```

SELECT 2
* find requested units and store them into memvars
LOCATE FOR serno = tserno
STORE unit1 TO tu1
STORE unit2 TO tu2
STORE unit3 TO tu3
STORE .F. TO found
STORE .T. TO requested

GO TOP
* find available units
DO WHILE .NOT. EOF() .AND. .NOT. found
  IF .NOT. DELETED()
    IF serno # tserno .AND. specialty = tspec,
      .AND. source = tsource
      STORE serno TO idno
      * unit available
      DO CASE
        CASE unitname = tu1
          * unit is the 1st requested
          STORE .T. TO found
          STORE unitname TO tunit
          STORE 0 TO tweight
        CASE unitname = tu2
          * unit is the 2nd requested
          STORE .T. TO found
          STORE unitname TO tunit
          STORE 1 TO tweight
        CASE unitname = tu3
          * unit is the 3rd requested
          STORE .T. TO found
          STORE unitname TO tunit
          STORE 2 TO tweight
        OTHERWISE
          * found available unit is not requested
          STORE .F. TO requested
      ENDCASE
    ENDIF
  SKIP
ENDDO

IF found
  * unit available is one of the requested units, mark
  * record for deletion, so that it will not be
  * encountered again
  LOCATE FOR serno = idno
  DELETE
  STORE .T. TO resolved

```

```

ELSE
  IF .NOT. requested
    * available unit is not requested
    * mark record for deletion
    LOCATE FOR serno = idno
    STORE unitname TO tunit
    STORE 3 TO tweight
    STORE .T. TO resolved
    DELETE
  ELSE
    @ 22,18 SAY 'OFFICER CANNOT BE ASSIGNED' GET tserno
  ENDIF
ENDIF
SELECT 1
FIND &tserno

IF resolved
  * update record in assigned file
  REPLACE unitname WITH tunit, asnweight WITH tweight
ELSE
  * No available unit found, delete created in assigned
  * file record. Officer will remain in the same unit
  DELETE
  STORE .T. TO deletion
ENDIF
SELECT 5
STORE .F. TO resolved
SKIP
ENDDO
ENDIF
IF deletion
  SELECT 1
  PACK
ENDIF

* update USERLOG file
SELECT 6
USE userid
LOCATE FOR password = UPPER(psw)
SELECT 7
USE userlog
APPEND BLANK
REPLACE username WITH F->username, task WITH 'ASSIGNMENTS',,
  progname WITH 'ASSIGN01', logdate WITH DATE(),,
  logtime WITH TIME()
CLOSE DATABASES
DELETE FILE file5.dbf
DELETE FILE file6.dbf
DELETE FILE file7.dbf
DELETE FILE file8.dbf
RETURN

```

***** PROGRAM MKAUXFL2 *****

* This program creates the temporary files required for the
* for the assignment processing of the 1st lieutenants

```
DO window
SELECT 1
USE serves
SELECT 2
USE officer
SELECT 3
USE requests
SELECT 2
@ 10,30 SAY 'FILE3 .....'
COPY TO file3 FOR rank = '02'
@ 10,45 SAY 'READY'
SELECT 1
@ 11,30 SAY 'FILE4 .....'
COPY TO file4 FOR enroldate <= CTOD('07/31/86') - 1065
@ 11,45 SAY 'READY'
SELECT 5
USE file4
SELECT 6
USE file3
@ 12,30 SAY 'FILE5 .....'
JOIN WITH E TO file5 FOR serno = E->serno ,
    FIELDS serno,rank,source,specialty,asnweight,marstat,,
    children,unitname,enroldate
@ 12,45 SAY 'READY'
SELECT 3
USE requests
SELECT 7
USE file5
@ 13,30 SAY 'FILE6 .....'
JOIN WITH C TO file6 FOR serno = C->serno ,
    FIELDS serno,rank,source,specialty,asnweight,marstat,,
    children,unitname,enroldate,unit1,unit2,unit3,submdate
@ 13,45 SAY 'READY'
CLOSE DATABASES
DELETE FILE file5.dbf
USE file6
COPY TO file5
@ 14,30 SAY 'FILE7 .....'
COPY TO file7
@ 14,45 SAY 'READY'
CLOSE DATABASES
DELETE FILE file3.dbf
DELETE FILE file4.dbf
RETURN
```

***** PROGRAM ASSIGN02 *****

* This program performs the assignments of the 2d lieutenants,
* and updates USERLOG file

* build required temporary files
DO mkauxfl1

* open required for the processing files
SELECT 1
USE assigned INDEX assigned
SELECT 2
USE officer

* display window on the screen
DO window1

* Assign officers recently graduated from MA.
* All of them are assigned to the AS for training

@ 5,20 SAY 'ASSIGNMENTS FOR THE NEW 2d LIEUTENANTS'
@ 6,20 SAY 'WHO HAVE RECENTLY GRADUATED FROM THE MA'
@ 7,20 SAY ' ARE BEING PROCESSED'

SELECT 2
LOCATE FOR rank = '01' .AND. source = 'MA',
.AND. YEAR(nomdate) = YEAR(DATE())
DO WHILE .NOT. EOF()
SELECT 1
APPEND BLANK
REPLACE serno WITH B->serno, rank WITH B->rank,
source WITH B->source, specialty WITH B->specialty
REPLACE unitname WITH 'AS', asgndate WITH CTOD('09/01/86'),
asnweight WITH 0
SELECT 2
CONTINUE
ENDDO
CLOSE DATABASES

* clear messages inside the window
@ 5,20 SAY '
@ 6,20 SAY '
@ 7,20 SAY '

* assign 2nd lieutenants completing their
* training in the Artillery School to ARTC

@ 5,23 SAY 'ASSIGNMENTS FOR THE 2d LIEUTENANTS'
@ 6,23 SAY ' GRADUATING FROM THE AS'
@ 7,23 SAY ' ARE BEING PROCESSED'

```

* open required for the processing files
SELECT 1
USE assigned INDEX assigned
SELECT 6
USE serves INDEX serves
LOCATE FOR unitname = 'AS' .AND. duty = 'TRAINEE'
DO WHILE .NOT. EOF()
    SELECT 1
    APPEND BLANK
    REPLACE serno WITH F->serno, unitname WITH 'ARTC',
           rank WITH '01', asgndate WITH CTOD('08/01/86')
    REPLACE source WITH 'MA', specialty WITH 'C',
           asnweight WITH 0
    SELECT 6
    CONTINUE
ENDDO

```

```

@ 5,20 SAY '
@ 6,20 SAY '
@ 7,20 SAY '

```

```

* assign 2nd lieutenants serving in the ARTC
* to the combat units in which they can serve

```

```

@ 5,23 SAY 'ASSIGNMENTS FOR THE 2d LIEUTENANTS'
@ 6,23 SAY ' WITH ORIGIN MA SERVING IN ARTC'
@ 7,23 SAY ' ARE BEING PROCESSED'
@ 8,23 SAY ' FIRST PASS'

```

```

* open required files
SELECT 3
USE requests INDEX requests
SELECT 4
USE unit INDEX unit
SELECT 5
USE unitorg INDEX unitorg
SELECT 8
USE file2
STORE .T. TO ok
DO WHILE .NOT. EOF()
    STORE ' ' TO tsource
    STORE serno TO tserno
    STORE rank TO trank
    STORE source TO tsource
    STORE specialty TO tspec
    STORE ' ' TO tunit, tu1, tu2, tu3
    STORE ' ' TO te1, te2, te3
    STORE ' ' TO tr1, tr2, tr3
    STORE 0 TO x1, x2, x3

```

```

* find what units the officer has requested
SELECT 3
FIND &tserno
STORE unit1 TO tu1
STORE unit2 TO tu2
STORE unit3 TO tu3

* find requested units in the unit file, and store the
* fields echelon and readiness into temporary memvars
SELECT 4
FIND &tu1
STORE echelon TO te1
STORE readiness TO tr1
FIND &tu2
STORE echelon TO te2
STORE readiness TO tr2
FIND &tu3
STORE echelon TO te3
STORE readiness TO tr3
SELECT 5
LOCATE FOR echelon = te1 .AND. readiness = tr1
STORE mac01 TO x1
LOCATE FOR echelon = te2 .AND. readiness = tr2
STORE mac01 TO x2
LOCATE FOR echelon = te3 .AND. readiness = tr3
STORE mac01 TO x3

* check if the officer can serve in the requested units
* according to the unit organization

IF ((x1 # 0) .AND. (x2 # 0) .AND. (x3 # 0))
  * requests are valid, check if 1st requested unit is
  * available
  SELECT 1
  LOCATE FOR rank = trank .AND. unitname = tu1

  IF EOF()
    * requested unit is available, unit is granted
    STORE tu1 TO tunit
    STORE 0 TO weight

  ELSE
    * 1st requested unit is not available
    * check if 2nd requested unit is available
    LOCATE FOR rank = trank .AND. unitname = tu2
    IF EOF()
      * requested unit is available, unit is granted
      STORE tu2 TO tunit
      STORE 1 TO weight

```



```

ELSE
  * 2nd requested unit is not available
  * check if 3rd requested unit is available
    LOCATE FOR rank = trunk .AND. unitname = tu3
    IF EOF()
      * requested unit is available, unit is granted
      STORE tu3 TO tunit
      STORE 2 TO weight
    ELSE
      * none of the requested units is available
      * assignment will be resolved in the 2nd pass
      STORE '*' TO tunit
      STORE 3 TO weight
      STORE .F. TO ok
    ENDIF
  ENDIF
ENDIF

* build assignments record for the officer
APPEND BLANK
REPLACE serno WITH tserno, unitname WITH tunit,
       rank WITH trunk, asgndate WITH CTOD('08/15/86')
REPLACE asnweight WITH weight, source WITH tsource ,
       specialty WITH tspec

ENDIF
SELECT 8
SKIP
ENDDO

* if there are deferred assignments from the first pass, try to
* resolve them
IF .NOT. ok
  SELECT 1
  @ 8,23 SAY '                SECOND PASS'

  * build file containing the deferred assignments
  COPY TO notok12 FOR unitname = '*' .AND. rank = '01',
       .AND. source = 'MA'
  SELECT 9
  USE notok12
  DO WHILE .NOT. EOF()
    STORE serno TO tserno
    STORE rank TO trunk
    SELECT 4
    GO TOP
    STORE .F. TO stop
  
```

```

* find units not requested by any officer in which a 2nd
* lieutenant can serve and assign non-assigned officers
* to them
DO WHILE .NOT. EOF() .AND. .NOT. stop
  IF category = 'CU'
    STORE ' ' TO tel
    STORE ' ' TO tr1
    STORE ' ' TO tunit
    STORE unitname TO tunit
    STORE echelon TO tel
    STORE readiness TO tr1
    SELECT 5
    SEEK tel + tr1
    IF .NOT. EOF()
      IF mac01 # 0
        SELECT 1
        LOCATE FOR rank=trank .AND. unitname= tunit
        IF EOF()
          FIND &tserno
          REPLACE unitname WITH tunit
          STORE .T. TO stop
        ENDIF
      ENDIF
    ENDIF
  ENDIF
  SELECT 4
  SKIP
ENDDO
IF .NOT. stop
  @ 22,18 SAY '*** INVALID REQUEST OR INCORRECT FILE'
ENDIF
SELECT 9
SKIP
ENDDO
ENDIF
CLOSE DATABASES

* delete non-required temporary files
DELETE FILE file1.dbf
DELETE FILE file2.dbf
DELETE FILE notok12.dbf

* assigne 2nd lieutenants whose source is NCOS
* to the units in which they can serve by combining
* requests and assignment weighs

@ 5,20 SAY '
@ 6,20 SAY '
@ 7,20 SAY '
@ 8,20 SAY '

```

```

@ 5,23 SAY 'ASSIGNMENTS FOR THE 2d LIEUTENANTS'
@ 6,23 SAY '          WHOSE SOURCE IS NCOS'
@ 7,23 SAY '          ARE BEING PROCESSED'

```

```

* open required for the processing files

```

```

SELECT 1
USE assigned INDEX assigned
SELECT 2
USE file5
SELECT 3
USE file6
SELECT 4
USE file7
STORE .T. TO ok
@ 8,23 SAY '
@ 8,23 SAY '          FIRST PASS'
STORE '08/15/86' TO tdate

```

```

* get each officer under assignment and determine the
* unit to be assigned or defer it for the second pass
DO WHILE .NOT. EOF()

```

```

* initialize temporary memvars
  STORE .F. TO granted
  STORE .T. TO preference
  STORE serno TO tserno
  STORE rank TO trank
  STORE source TO tsource
  STORE specialty TO tspec
  STORE asnweight TO weight
  STORE marstat TO marst
  STORE ' ' TO tunit,tu1,tu2,tu3
  STORE unit1 TO tu1
  STORE unit2 TO tu2
  STORE unit3 TO tu3
  STORE .F. TO resolved
  STORE 1 TO loopcnt

```

```

  * determine which of the requested units
  * will be examined next (1st, 2nd, or 3rd)
  DO WHILE .NOT. resolved .AND. loopcnt < 4
    IF loopcnt = 1
      STORE tu1 TO tu
    ELSE
      IF loopcnt = 2
        STORE tu2 TO tu
      ELSE
        STORE tu3 TO tu
      ENDIF
    ENDIF
  ENDIF

```

```

* check if requested by the officer unit is available,
* i.e., examine if there exists an officer under
* assignment whose present unit is the requested unit
SELECT 3
LOCATE FOR specialty = tspec .AND. unitname = tu,
      .AND. serno # tserno .AND. .NOT. DELETED()
IF .NOT. EOF()
  STORE serno TO idno
  SELECT 2
  LOCATE FOR serno = idno .AND. .NOT. DELETED()

  IF .NOT. EOF()
    * someone else from the requested unit is to be moved
    * check if someone else has requested the same unit
    SELECT 3
    STORE .F. TO done
    DO WHILE .NOT. EOF() .AND. .NOT. done
      IF .NOT. DELETED()
        IF specialty = tspec .AND. serno # tserno
          IF unit1=tu .OR. unit2=tu .OR. unit3=tu
            STORE .T. TO done
          ENDIF
        ENDIF
      ENDIF
      SKIP
    ENDDO

    IF .NOT. done
      * nobody else requests the same unit, requested
      * unit is granted, record is marked for deletion
      STORE tu TO tunit
      DO CASE
        CASE loopcnt = 1
          STORE 0 TO tweight
        CASE loopcnt = 2
          STORE 1 TO tweight
        CASE loopcnt = 3
          STORE 2 TO tweight
      ENDCASE
      STORE .T. TO granted
      LOCATE FOR serno = tserno
      DELETE
      SELECT 2
      LOCATE FOR serno = idno
      DELETE
      STORE .T. TO resolved
    
```

```

ELSE
* somebody else requests the same unit, check
* assignment weights to see whom will be given
* the preference to
DO WHILE ((.NOT. EOF()) .AND. (preference))
  IF .NOT. DELETED()
    IF serno # tserno .AND. specialty = tspec
      IF unit1=tu .OR. unit2=tu,
        .OR. unit3=tu
        IF asnweight > weight
          STORE .F. TO preference
        ELSE
          IF asnweight = weight
            IF marst = 'S',
              .AND. marstat # 'S'
              STORE .F. TO preference
            ENDIF
          ENDIF
        ENDIF
      ENDIF
    ENDIF
  ENDIF
  SKIP
ENDDO

IF preference
* officer requested this unit has the
* preference, unit is granted, record
* is marked for deletion
STORE tu TO tunit
DO CASE
  CASE loopcnt = 1
    STORE 0 TO tweight
  CASE loopcnt = 2
    STORE 1 TO tweight
  CASE loopcnt = 3
    STORE 2 TO tweight
ENDCASE
STORE .T. TO granted
LOCATE FOR serno = tserno
DELETE
SELECT 2
LOCATE FOR serno = idno
DELETE
STORE .T. TO resolved
ENDIF
ENDIF
ENDIF
STORE loopcnt + 1 TO loopcnt
ENDDO

```

```

IF .NOT. resolved
* unresolved assignment, defer it for the next pass
  STORE '*' TO tunit
  STORE 3 TO tweight
  STORE .F. TO ok
ENDIF

* build record in the file containing the assignments
SELECT 1
APPEND BLANK
REPLACE serno WITH tserno, rank WITH trank,,
      source WITH tsource, specialty WITH tspec
REPLACE unitname WITH tunit, asnweight WITH tweight,,
      asgndate WITH CTOD(tdate)
SELECT 4
SKIP
ENDDO

* second pass

IF .NOT. ok
* unresolved assignments exist
  @ 8,23 SAY '          SECOND PASS'
  STORE .F. TO resolved

  SELECT 1
  * build temporary file containing unresolved assignments
  COPY TO file8 FOR unitname = '*' .AND. source = 'NCOS'
  SELECT 5
  USE file8
  STORE .F. TO deletion

  * get one record at a time and try to satisfy request
  DO WHILE .NOT. EOF()
    STORE serno TO tserno
    STORE specialty TO tspec
    STORE '      ' TO tunit

    SELECT 2
    * find requested units and store them into memvars
    LOCATE FOR serno = tserno
    STORE unit1 TO tu1
    STORE unit2 TO tu2
    STORE unit3 TO tu3
    STORE .F. TO found
    STORE .T. TO requested
    GO TOP

```

```

* find available units
DO WHILE .NOT. EOF() .AND. .NOT. found
  IF .NOT. DELETED()
    IF serno # tserno .AND. specialty = tspec
      STORE serno TO idno
      * unit available
      DO CASE
        CASE unitname = tu1
          * unit is the 1st requested
          STORE .T. TO found
          STORE unitname TO tunit
          STORE 0 TO tweight
        CASE unitname = tu2
          * unit is the 2nd requested
          STORE .T. TO found
          STORE unitname TO tunit
          STORE 1 TO tweight
        CASE unitname = tu3
          * unit is the 3rd requested
          STORE .T. TO found
          STORE unitname TO tunit
          STORE 2 TO tweight
        OTHERWISE
          * found available unit is not requested
          STORE .F. TO requested
      ENDCASE
    ENDIF
  ENDIF
  SKIP
ENDDO

IF found
  * unit available is one of the requested, mark record
  * for deletion, so that it will not be encountered again
  LOCATE FOR serno = idno
  DELETE
  STORE .T. TO resolved
ELSE
  IF .NOT. requested
    * available unit is not requested
    * mark record for deletion
    LOCATE FOR serno = idno
    STORE unitname TO tunit
    STORE 3 TO tweight
    STORE .T. TO resolved
    DELETE
  ELSE
    @ 22,18 SAY 'OFFICER CANNOT BE ASSIGNED' GET tserno
  ENDIF
ENDIF

```

```

SELECT 1
FIND &tserno

IF resolved
* update record in assigned file
  REPLACE unitname WITH tunit, asnweight WITH tweight
ELSE

* no available unit found, delete created in assigned
* file record officer will remain in the same unit
  DELETE
  STORE .T. TO deletion
ENDIF
SELECT 5
STORE .F. TO resolved
SKIP
ENDDO
ENDIF
IF deletion
  SELECT 1
  PACK
ENDIF

* update USERLOG file
SELECT 6
USE userid
LOCATE FOR password = UPPER(psw)
SELECT 7
USE userlog
APPEND BLANK
REPLACE username WITH F->username, task WITH 'ASSIGNMENTS',,
        progame WITH 'ASSIGN02', logdate WITH DATE(),,
        logtime WITH TIME()
CLOSE DATABASES
DELETE FILE file5.dbf
DELETE FILE file6.dbf
DELETE FILE file7.dbf
DELETE FILE file8.dbf
RETURN

```


***** PROGRAM MKAUXFL1 *****

* This program creates the temporary files required for the
* assignment processing of the 2nd lieutenants

```
DO window
SELECT 1
USE serves
SELECT 2
USE officer
SELECT 3
USE requests
SELECT 1
@ 8,30 SAY 'FILE1 .....'
COPY TO file1 FOR unitname = 'ARTC' .AND. duty = 'P/CDR'
@ 8,45 SAY 'READY'
SELECT 4
USE file1
@ 9,30 SAY 'FILE2 .....'
JOIN WITH B TO file2 FOR serno=B->serno .AND. B->source='MA',
    FIELDS serno, rank, source, specialty, unitname
@ 9,45 SAY 'READY'
SELECT 2
@ 10,30 SAY 'FILE3 .....'
COPY TO file3 FOR rank = 'O1' .AND. source = 'NCOS'
@ 10,45 SAY 'READY'
SELECT 1
@ 11,30 SAY 'FILE4 .....'
COPY TO file4 FOR enroldate <= CTOD('07/31/86') - 1065
@ 11,45 SAY 'READY'
SELECT 5
USE file4
SELECT 6
USE file3
@ 12,30 SAY 'FILE5 .....'
JOIN WITH E TO file5 FOR serno = E->serno ,
    FIELDS serno,rank,source,specialty,asnweight,marstat,,
    children, unitname, enroldate
@ 12,45 SAY 'READY'
SELECT 3
USE requests
SELECT 7
USE file5
@ 13,30 SAY 'FILE6 .....'
JOIN WITH C TO file6 FOR serno = C->serno ,
    FIELDS serno,rank,source,specialty,asnweight,marstat,,
    children,unitname,enroldate,unit1,unit2,unit3,submdate
@ 13,45 SAY 'READY'
CLOSE DATABASES
DELETE FILE file5.dbf
USE file6
```

```
COPY TO file5
@ 14,30 SAY 'FILE7 ..... '
COPY TO file7
@ 14,45 SAY 'READY'
CLOSE DATABASES
DELETE FILE file3.dbf
DELETE FILE file4.dbf
RETURN
```

***** PROGRAM ASSIGN03 *****

* This performs the assignments of the captains, and updates
* USERLOG file

* build required temporary files
DO mkauxfl3

* display window on the screen
DO window1

* assign captains to units in which they can serve
* by combining requests and assignment weights

@ 5,23 SAY ' ASSIGNMENTS FOR THE CAPTAINS'
@ 6,23 SAY ' ARE BEING PROCESSED'

* open required for the processing files

SELECT 1
USE assigned INDEX assigned

SELECT 2
USE file5

SELECT 3
USE file6

SELECT 4
USE file7

STORE .T. TO ok

@ 8,23 SAY '

@ 8,23 SAY ' FIRST PASS'

STORE '08/15/86' TO tdate

* get each officer under assignment and determine the
* unit to be assigned or defer it for the second pass

DO WHILE .NOT. EOF()

* initialize memvars

STORE .F. TO granted

STORE .T. TO preference

STORE serno TO tserno

STORE rank TO trank

STORE source TO tsource

STORE specialty TO tspec

STORE asnweight TO weight

STORE marstat TO marst

STORE ' ' TO tunit,tu1,tu2,tu3

STORE unit1 TO tu1

STORE unit2 TO tu2

STORE unit3 TO tu3

STORE .F. TO resolved

STORE 1 TO loopcnt

```

* determine which of the requested units
* will be examined next (1st, 2nd, or 3rd)
DO WHILE .NOT. resolved .AND. loopcnt < 4
  IF loopcnt = 1
    STORE tu1 TO tu
  ELSE
    IF loopcnt = 2
      STORE tu2 TO tu
    ELSE
      STORE tu3 TO tu
    ENDIF
  ENDIF
ENDIF

* check if requested by the officer unit is available
SELECT 3
LOCATE FOR specialty=tspec .AND. source=tsource .AND.,
  unitname=tu .AND. serno#tserno .AND. .NOT. DELETED()
IF .NOT. EOF()
  STORE serno TO idno
  SELECT 2
  LOCATE FOR serno = idno .AND. .NOT. DELETED()

  IF .NOT. EOF()
    * requested unit available, check if someone else
    * requests the same unit
    SELECT 3
    STORE .F. TO done
    DO WHILE .NOT. EOF() .AND. .NOT. done
      IF .NOT. DELETED()
        IF specialty = tspec .AND. serno # tserno,
          .AND. source = tsource
          IF unit1=tu .OR. unit2=tu .OR. unit3=tu
            STORE .T. TO done
          ENDIF
        ENDIF
      ENDIF
    ENDIF
    SKIP
  ENDDO

  IF .NOT. done
    * nobody else requests the same unit, requested
    * unit is granted, record is marked
    STORE tu TO tunit
    DO CASE
      CASE loopcnt = 1
        STORE 0 TO tweight
      CASE loopcnt = 2
        STORE 1 TO tweight
      CASE loopcnt = 3
        STORE 2 TO tweight
    ENDCASE
  
```

```

STORE .T. TO granted
LOCATE FOR serno = tserno
DELETE
SELECT 2
LOCATE FOR serno = idno
DELETE
STORE .T. TO resolved

```

```

ELSE
* somebody else requests the same unit, check
* assignment weights to see whom will be given
* the preference to
DO WHILE ((.NOT. EOF()) .AND. (preference))
  IF .NOT. DELETED()
    IF serno # tserno .AND. specialty=tspec,
      .AND. source = tsource
      IF unit1 = tu .OR. unit2 = tu,
        .OR. unit3 = tu
        IF asnweight > weight
          STORE .F. TO preference
        ELSE
          IF asnweight = weight
            IF marst = 'S',
              .AND. marstat # 'S'
              STORE .F. TO preference
            ENDIF
          ENDIF
        ENDIF
      ENDIF
    ENDIF
  ENDIF
  SKIP
ENDDO

```

```

IF preference
* officer requested this unit has the
* preference, unit is granted, record
* is marked for deletion
STORE tu TO tunit
DO CASE
  CASE loopcnt = 1
    STORE 0 TO tweight
  CASE loopcnt = 2
    STORE 1 TO tweight
  CASE loopcnt = 3
    STORE 2 TO tweight
ENDCASE
STORE .T. TO granted
LOCATE FOR serno = tserno
DELETE

```

```

                SELECT 2
                LOCATE FOR serno = idno
                DELETE
                STORE .T. TO resolved
            ENDIF
        ENDIF
    ENDIF
    ENDIF
    STORE loopcnt + 1 TO loopcnt
ENDDO

IF .NOT. resolved
* unresolved assignment, defer it for the next pass
    STORE '*' TO tunit
    STORE 3 TO tweight
    STORE .F. TO ok
ENDIF

* build record in the file containing the assignments
SELECT 1
APPEND BLANK
REPLACE serno WITH tserno, rank WITH trank,,
        source WITH tsource, specialty WITH tspec
REPLACE unitname WITH tunit, asnweight WITH tweight,,
        asgndate WITH CTOD(tdate)
SELECT 4
SKIP
ENDDO

* second pass

IF .NOT. ok
* unresolved assignments exist
    @ 8,23 SAY '                SECOND PASS'
    STORE .F. TO resolved
    STORE .F. TO deletion
    SELECT 1

* build temporary file containing unresolved assignments
COPY TO file8 FOR unitname = '*' .AND. rank = '03'
SELECT 5
USE file8

* get one record at a time and try to satisfy request
DO WHILE .NOT. EOF()
    * initialize memvars
    STORE ' ' TO tsource
    STORE serno TO tserno
    STORE specialty TO tspec
    STORE source TO tsource
    STORE ' ' TO tunit

```

```

SELECT 2
* find requested units and store them into memvars
LOCATE FOR serno = tserno
STORE unit1 TO tu1
STORE unit2 TO tu2
STORE unit3 TO tu3
STORE .F. TO found
STORE .T. TO requested
GO TOP

* find available units
DO WHILE .NOT. EOF() .AND. .NOT. found
  IF .NOT. DELETED()
    IF serno # tserno .AND. specialty = tspec,
      .AND. source = tsource
      STORE serno TO idno
      * unit available
      DO CASE
        CASE unitname = tu1
          * unit is the 1st requested
          STORE .T. TO found
          STORE unitname TO tunit
          STORE 0 TO tweight
        CASE unitname = tu2
          * unit is the 2nd requested
          STORE .T. TO found
          STORE unitname TO tunit
          STORE 1 TO tweight
        CASE unitname = tu3
          * unit is the 3rd requested
          STORE .T. TO found
          STORE unitname TO tunit
          STORE 2 TO tweight
        OTHERWISE
          * found available unit is not requested
          STORE .F. TO requested
      ENDCASE
    ENDIF
  ENDIF
  SKIP
ENDDO

IF found
  * unit available is one of the requested units, mark
  *   record for deletion, so that it will not be
  *   encountered again
  LOCATE FOR serno = idno
  DELETE
  STORE .T. TO resolved

```

```

ELSE
  IF .NOT. requested
    * available unit is not requested
    * mark record for deletion
    LOCATE FOR serno = idno
    STORE unitname TO tunit
    STORE 3 TO tweight
    STORE .T. TO resolved
    DELETE
  ELSE
    @ 22,18 SAY 'OFFICER CANNOT BE ASSIGNED' GET tserno
  ENDIF
ENDIF

SELECT 1
FIND &tserno
IF resolved
  * update record in assigned file
  REPLACE unitname WITH tunit, asnweight WITH tweight

ELSE
  * no available unit found, delete created in assigned
  * file record officer, will remain in the same unit
  DELETE
  STORE .T. TO deletion
ENDIF
SELECT 5
STORE .F. TO resolved
SKIP
ENDDO
ENDIF
IF deletion
  SELECT 1
  PACK
ENDIF
* update USERLOG file
SELECT 6
USE userid
LOCATE FOR password = UPPER(psw)
SELECT 7
USE userlog
APPEND BLANK
REPLACE username WITH F->username, task WITH 'ASSIGNMENTS',,
  progname WITH 'ASSIGN03', logdate WITH DATE(),,
  logtime WITH TIME()
CLOSE DATABASES
DELETE FILE file5.dbf
DELETE FILE file6.dbf
DELETE FILE file7.dbf
DELETE FILE file8.dbf
RETURN

```


***** PROGRAM MKAUXFL3 *****

* This program creates auxiliary files required for the
 * assignment processing of the captains

```
DO window
SELECT 1
USE serves
SELECT 2
USE officer
SELECT 3
USE requests
SELECT 2
@ 10,30 SAY 'FILE3 .....
```

COPY TO file3 FOR rank = '03'

```
@ 10,45 SAY 'READY'
SELECT 1
@ 11,30 SAY 'FILE4 .....
```

COPY TO file4 FOR enroldate <= CTOD('07/31/86') - 1065

```
@ 11,45 SAY 'READY'
SELECT 5
USE file4
SELECT 6
USE file3
@ 12,30 SAY 'FILE5 .....
```

JOIN WITH E TO file5 FOR serno = E->serno ,
 FIELDS serno,rank,source,specialty,asnweight,marstat,,
 children,'unitname, enroldate

```
@ 12,45 SAY 'READY'
SELECT 7
USE file5
@ 13,30 SAY 'FILE6 .....
```

JOIN WITH C TO file6 FOR serno = C->serno ,
 FIELDS serno,rank,source,specialty,asnweight,marstat,,
 children,unitname,enroldate,unit1,unit2,unit3,submdate

```
@ 13,45 SAY 'READY'
CLOSE DATABASES
DELETE FILE file5.dbf
USE file6
COPY TO file5
@ 14,30 SAY 'FILE7 .....
```

COPY TO file7

```
@ 14,45 SAY 'READY'
CLOSE DATABASES
DELETE FILE file3.dbf
DELETE FILE file4.dbf
RETURN
```

***** PROGRAM ASSIGN04 *****

* This program performs the assignments of the majors

DO mkauxfl4
DO window1

* assign majors graduating from WC to artillery staffs
* by combining requests and assignment weights

@ 5,23 SAY ' ASSIGNMENTS FOR THE MAJORS'
@ 6,23 SAY ' GRADUATING FROM THE WAR COLLEGE'
@ 7,23 SAY ' ARE BEING PROCESSED'
@ 8,23 SAY ' FIRST PASS'
STORE '08/05/86' TO tdate

* open required for the processing files

SELECT 1
USE assigned INDEX assigned
SELECT 2
USE staffs
SELECT 3
USE file3
SELECT 4
USE file4
SELECT 3
STORE 0 TO cnt
STORE .T. TO found

DO WHILE found

IF .NOT. DELETED()
STORE ' ' TO tunit,tu1,tu2,tu3
STORE serno TO tserno
STORE source TO tsource
STORE specialty TO tspec
STORE asnweight TO weight
STORE unit1 TO tu1
STORE unit2 TO tu2
STORE unit3 TO tu3

* find requested units and store name, echelon,
* and readiness into temporary memvars
STORE 1 TO loopcnt
STORE .F. TO resolved
STORE .T. TO preference

```

DO WHILE .NOT. resolved .AND. loopcnt < 4
  IF loopcnt = 1
    STORE tu1 TO tu
  ELSE
    IF loopcnt = 2
      STORE tu2 TO tu
    ELSE
      STORE tu3 TO tu
    ENDIF
  ENDIF
  SELECT 1
  LOCATE FOR serno # tserno .AND. unitname = tu

  IF EOF()
    * requested unit is available
    * check if somebody else requests the same unit
    SELECT 4
    DO WHILE .NOT. EOF() .AND. preference
      IF .NOT. DELETED() .AND. serno # tserno
        IF unit1=tu .OR. unit2=tu .OR. unit3=tu
          IF asnweight > weight
            STORE .F. TO preference
          ELSE
            IF asnweight = weight
              IF marst = 'S' .AND. marstat # 'S'
                STORE .F. TO preference
              ENDIF
            ENDIF
          ENDIF
        ENDIF
      ENDIF
      SKIP
    ENDDO

    IF preference
      * officer requested this unit has the preference,
      * unit is granted, record is marked for deletion
      STORE tu TO tunit

      DO CASE
        CASE loopcnt = 1
          STORE 0 TO tweight
        CASE loopcnt = 2
          STORE 1 TO tweight
        CASE loopcnt = 3
          STORE 2 TO tweight
      ENDCASE
    
```

```

        LOCATE FOR serno = tserno
        DELETE
        SELECT 1
        FIND &tserno
        REPLACE unitname WITH tunit,,
                asnweight WITH tweight,,
                asgndate WITH CTOD(tdate)
        STORE cnt + 1 TO cnt
        STORE .T. TO resolved
        SELECT 3
        DELETE
        SELECT 2
        LOCATE FOR unitname=tunit .AND. .NOT. DELETED()
        DELETE
    ENDIF
    ENDIF
    STORE loopcnt + 1 TO loopcnt
ENDDO

ENDIF
SELECT 3
SKIP
IF EOF()
    GO TOP
    IF cnt # 0
        STORE 0 TO cnt
    ELSE
        STORE .F. TO found
    ENDIF
ENDIF

ENDIF

ENDDO
@ 8,23 SAY '                SECOND PASS'
SELECT 3
GO TOP

DO WHILE .NOT. EOF()
    IF .NOT. DELETED()
        STORE serno TO tserno
        STORE unit1 TO tu1
        STORE unit2 TO tu2
        STORE unit3 TO tu3
        STORE .T. TO found
        SELECT 2
        LOCATE FOR unitname = tu1 .AND. .NOT. DELETED()
        IF .NOT. EOF()
            STORE 0 TO tweight
            STORE tu1 TO tunit
            DELETE
        
```

```

ELSE
  LOCATE FOR unitname = tu2 .AND. .NOT. DELETED()
  IF .NOT. EOF()
    STORE 1 TO tweight
    STORE tu2 TO tunit
    DELETE
  ELSE
    LOCATE FOR unitname = tu3 .AND. .NOT. DELETED()
    IF .NOT. EOF()
      STORE 2 TO tweight
      STORE tu3 TO tunit
      DELETE
    ELSE
      LOCATE FOR .NOT. DELETED()
      IF .NOT. EOF()
        STORE 3 TO tweight
        STORE unitname TO tunit
        DELETE
      ELSE
        @ 23,18 SAY '*** ERROR'
        STORE .F. TO found
      ENDIF
    ENDIF
  ENDIF
ENDIF
ENDIF
ENDIF
IF found
  SELECT 1
  FIND &tserno
  REPLACE unitname WITH tunit, asnweight WITH tweight,,
    asgndate WITH CTOD(tdate)
ENDIF

SELECT 3
DELETE
SKIP
ENDDO

CLOSE DATABASES
DELETE file file3.dbf
DELETE file file4.dbf
DELETE FILE staffs.dbf

```

```

* assign majors to units and staffs in which they can
* serve by combining requests and assignment weights
DO window1

```

```

@ 5,23 SAY '      ASSIGNMENTS FOR THE MAJORS'
@ 6,23 SAY 'SERVING IN STAFFS AND COMBAT UNITS'
@ 7,23 SAY '      ARE BEING PROCESSED'

```

```

* open required for the processing files

```

```

SELECT 1
USE assigned INDEX assigned
SELECT 2
USE file5
SELECT 3
USE file6
SELECT 4
USE selected INDEX selected
SELECT 5
USE file7
STORE .T. TO ok

```

```

@ 8,23 SAY '
@ 8,23 SAY '      FIRST PASS'
STORE '08/15/86' TO tdate

```

```

* get each officer under assignment and determine the
* unit to be assigned or defer it for the second pass

```

```

DO WHILE .NOT. EOF()
  STORE .F. TO granted
  STORE .T. TO preference
  STORE serno TO tserno
  STORE rank TO trank
  STORE source TO tsource
  STORE specialty TO tspec
  STORE asnweight TO weight
  STORE marstat TO marst
  STORE '      ' TO tunit,tu1,tu2,tu3
  SELECT 4
  FIND &tserno

```

```

  IF .NOT. EOF()
    STORE schoolname TO tunit
    STORE 0 TO tweight
    SELECT 3
    LOCATE FOR serno = tserno
    DELETE

```

```

  ELSE
    SELECT 5
    STORE unit1 TO tu1
    STORE unit2 TO tu2
    STORE unit3 TO tu3
    STORE .F. TO resolved
    STORE 1 TO loopcnt

```

```

* determine which of the requested units
* will be examined next (1st, 2nd, or 3rd)
DO WHILE .NOT. resolved .AND. loopcnt < 4
  IF loopcnt = 1
    STORE tu1 TO tu
  ELSE
    IF loopcnt = 2
      STORE tu2 TO tu
    ELSE
      STORE tu3 TO tu
    ENDIF
  ENDIF

* check if requested unit is available,
SELECT 3
LOCATE FOR specialty = tspec .AND. source = tsource,
      .AND. unitname = tu .AND. serno # tserno,
      .AND. .NOT. DELETED()
IF .NOT. EOF()
  STORE serno TO idno
  SELECT 2
  LOCATE FOR serno = idno .AND. .NOT. DELETED()

  IF .NOT. EOF()
    * someone else from the requested unit is to be
    * moved, check if someone else has requested the
    * same unit
    SELECT 3
    STORE .F. TO done

    DO WHILE .NOT. EOF() .AND. .NOT. done
      IF .NOT. DELETED()
        IF specialty=tspec .AND. serno # tserno,
          .AND. source = tsource
          IF unit1 = tu .OR. unit2 = tu,
            .OR. unit3 = tu
            STORE .T. TO done
          ENDIF
        ENDIF
      ENDIF
      SKIP
    ENDDO

    IF .NOT. done
      * nobody else requests the same unit, requested
      * unit is granted, record is marked for deletion
      STORE tu TO tunit
    
```

```

DO CASE
  CASE loopcnt = 1
    STORE 0 TO tweight
  CASE loopcnt = 2
    STORE 1 TO tweight
  CASE loopcnt = 3
    STORE 2 TO tweight
ENDCASE

STORE .T. TO granted
LOCATE FOR serno = tserno
DELETE
SELECT 2
LOCATE FOR serno = idno
DELETE
STORE .T. TO resolved
ELSE
  * somebody else requests the same unit, check
  * assignment weights to see whom will be given
  * the preference to
  DO WHILE ((.NOT. EOF()) .AND. (preference))
    IF .NOT. DELETED()
      IF serno # tserno,
        .AND. specialty = tspec,
        .AND. source = tsource
        IF unit1 = tu .OR. unit2 = tu,
          .OR. unit3 = tu
          IF asnweight > weight
            STORE .F. TO preference
          ELSE
            IF asnweight = weight
              IF marst = 'S',
                .AND. marstat # 'S'
                STORE .F. TO preference
            ENDIF
          ENDIF
        ENDIF
      ENDIF
    ENDIF
  ENDIF
  SKIP
ENDDO

IF preference
  * officer requested this unit has the
  * preference, unit is granted, record
  * is marked for deletion
  STORE tu TO tunit

```



```

DO CASE
  CASE loopcnt = 1
    STORE 0 TO tweight
  CASE loopcnt = 2
    STORE 1 TO tweight
  CASE loopcnt = 3
    STORE 2 TO tweight
ENDCASE

STORE .T. TO granted
LOCATE FOR serno = tserno
DELETE
SELECT 2
LOCATE FOR serno = idno
DELETE
STORE .T. TO resolved
ENDIF
ENDIF
ENDIF
STORE loopcnt + 1 TO loopcnt
ENDDO

IF .NOT. resolved
  * unresolved assignment, defer it for the next pass
  STORE '*' TO tunit
  STORE 3 TO tweight
  STORE .F. TO ok
ENDIF
ENDIF

* build record in the file containing the assignments
SELECT 1
APPEND BLANK
REPLACE serno WITH tserno, rank WITH trunk,,
       source WITH tsource, specialty WITH tspec
REPLACE unitname WITH tunit, asnweight WITH tweight,,
       asgndate WITH CTOD(tdate)
SELECT 5
SKIP
ENDDO

```

```

* second pass

IF .NOT. ok
* unresolved assignments exist
  SELECT 6
  USE unit INDEX unit
  @ 8,23 SAY '          SECOND PASS'
  STORE .F. TO resolved
  STORE .F. TO deletion

  SELECT 1
  * build temporary file containing unresolved assignments
  COPY TO file8 FOR unitname = '*' .AND. rank = '04'
  SELECT 5
  USE file8

  * get one record at a time and try to satisfy request
  DO WHILE .NOT. EOF()
    STORE ' ' TO tsource
    STORE serno TO tserno
    STORE specialty TO tspec
    STORE source TO tsource
    STORE ' ' TO tunit

    SELECT 2
    * find requested units and store them into memvars
    LOCATE FOR serno = tserno
    STORE ' ' TO tu
    STORE wcfinished TO twc
    STORE unit1 TO tu1
    STORE unit2 TO tu2
    STORE unit3 TO tu3
    STORE .F. TO found
    STORE .T. TO requested
    GO TOP

    * find available units
    DO WHILE .NOT. EOF() .AND. .NOT. found
      IF .NOT. DELETED()
        IF serno # tserno .AND. specialty = tspec,
          .AND. source = tsource
          STORE serno TO idno
          * unit available

```

```

DO CASE
CASE unitname = tu1
* unit is the 1st requested
STORE .T. TO found
STORE unitname TO tunit
STORE 0 TO tweight
CASE unitname = tu2
* unit is the 2nd requested
STORE .T. TO found
STORE unitname TO tunit
STORE 1 TO tweight
CASE unitname = tu3
* unit is the 3rd requested
STORE .T. TO found
STORE unitname TO tunit
STORE 2 TO tweight
OTHERWISE
* found available unit is not requested
IF unitname = 'OAB'
IF twc
STORE .F. TO requested
ENDIF
ELSE
STORE unitname TO tu
SELECT 6
FIND &tu
IF category = 'S'
IF twc
STORE .F. TO requested
ENDIF
ELSE
STORE .F. TO requested
ENDIF
ENDIF
SELECT 2
ENDCASE
ENDIF
ENDIF
SKIP
ENDDO

IF found
* unit available is one of the requested units, mark
* record for deletion, so that it will not be
* encountered again
LOCATE FOR serno = idno
DELETE
STORE .T. TO resolved

```

```

ELSE
  IF .NOT. requested
    * available unit is not requested
    * mark record for deletion
    LOCATE FOR serno = idno
    STORE unitname TO tunit
    STORE 3 TO tweight
    STORE .T. TO resolved
    DELETE
  ELSE
    @ 22,18 SAY 'OFFICER CANNOT BE ASSIGNED' GET tserno
  ENDIF
ENDIF
SELECT 1
FIND &tserno
IF resolved
  * update record in assigned file
  REPLACE unitname WITH tunit, asnweight WITH tweight
ELSE
  * no available unit found, delete created in assigned
  * file record. Officer will remain in the same unit
  DELETE
  STORE .T. TO deletion
ENDIF
SELECT 5
STORE .F. TO resolved
SKIP
ENDDO

ENDIF
IF deletion
  SELECT 1
  PACK
ENDIF

* update USERLOG file
SELECT 6
USE userid
LOCATE FOR password = UPPER(psw)
SELECT 7
USE userlog
APPEND BLANK
REPLACE username WITH F->username, task WITH 'ASSIGNMENTS',,
      progname WITH 'ASSIGN04', logdate WITH DATE(),,
      logtime WITH TIME()
CLOSE DATABASES
DELETE FILE file5.dbf
DELETE FILE file6.dbf
DELETE FILE file7.dbf
DELETE FILE file8.dbf
RETURN

```

***** PROGRAM MKAUXFL4 *****

* This program builds the required auxiliary files for the
* processing of the assignments of the majors

DO window

* build file containing personal data and
* requests of the graduating from WC majors
@ 8,30 SAY 'FILE3

SELECT 1

USE officer

SELECT 2

USE serves

COPY TO file1 FOR unitname = 'WC'

SELECT 3

USE file1

JOIN WITH A TO file2 FOR serno = A->serno ,

FIELDS serno,source,specialty,marstat,unitname,asnweight

SELECT 4

USE requests

SELECT 5

USE file2

JOIN WITH D TO file3 FOR serno = D->serno ,

FIELDS serno,source,specialty,unitname,unit1,unit2,unit3,,
asnweight,marstat,wcfinished

@ 8,45 SAY 'READY'

SELECT 6

USE file3

@ 9,30 SAY 'FILE4

COPY TO file4

@ 9,45 SAY 'READY'

CLOSE DATABASES

DELETE FILE file1.dbf

DELETE FILE file2.dbf

SELECT 7

USE assigned INDEX assigned

@ 10,30 SAY 'ASSIGNED

APPEND FROM file3

GO TOP

DO WHILE .NOT. EOF()

IF rank = '04'

REPLACE unitname WITH '*',,,

asgndate WITH CTOD('08/01/86'),,

asnweight WITH 0

ENDIF

SKIP

ENDDO

@ 10,45 SAY 'READY'

```

SELECT 8
USE unit
@ 11,30 SAY 'STAFFS .....
```

COPY TO staffs FOR category = 'S ' .AND. unitname # 'ABD/HAGS'

```

CLOSE DATABASES
SELECT 2
USE staffs
APPEND FROM unit FOR echelon = 'AC/AC'
@ 11,45 SAY 'READY'
CLOSE DATABASES
* build file containing personal data and requests
*   for the rest of majors under assignment
SELECT 1
USE serves
SELECT 2
USE officer
@ 12,30 SAY 'FILES .....
```

COPY TO file1 FOR rank = '04'

```

SELECT 1
COPY TO file2 FOR DATE() - enroldate >= 600,
      .AND. unitname # 'WC
SELECT 3
USE file2
SELECT 4
USE file1
JOIN WITH C TO file5 FOR serno = C->serno ,
      FIELDS serno,rank,source,specialty,asnweight,marstat,,
              unitname
@ 12,45 SAY 'READY'
CLOSE DATABASES
DELETE FILE file1.dbf
DELETE FILE file2.dbf
SELECT 5
USE requests
SELECT 6
USE file5
@ 13,30 SAY 'FILE6 .....
```

JOIN WITH E TO file6 FOR serno = E->serno,

```

      FIELDS serno,rank,source,specialty,asnweight,marstat,,
              unitname,unit1,unit2,unit3,wcfinished
CLOSE DATABASES
DELETE FILE file5.dbf
SELECT 7
USE file6
COPY TO file5
@ 13,45 SAY 'READY'
@ 14,30 SAY 'FILE7 .....
```

COPY TO file7

```

@ 14,45 SAY 'READY'
CLOSE DATABASES
RETURN
```

***** PROGRAM ASSIGN05 *****

* This program performs the assignments of the lieutenant
* colonels

* build necessary auxiliary files
DO mkauxf15

* assign lieutenant colonels to units and staffs in which
* they can serve by combining requests and assignment weights

DO window1
@ 5,21 SAY 'ASSIGNMENTS FOR THE LIEUTENANT COLONELS'
@ 6,21 SAY ' ARE BEING PROCESSED'

* open required for the processing files

SELECT 1
USE assigned INDEX assigned
SELECT 2
USE file5
SELECT 3
USE file6
SELECT 4
USE selected INDEX selected
SELECT 5
USE file7
STORE .T. TO ok
@ 8,23 SAY '
@ 8,23 SAY ' FIRST PASS'
STORE '08/15/86' TO tdate

* get each officer under assignment and determine the
* unit to be assigned or defer it for the second pass

DO WHILE .NOT. EOF()
STORE .F. TO granted
STORE .T. TO preference
STORE serno TO tserno
STORE rank TO trank
STORE source TO tsource
STORE specialty TO tspec
STORE asnweight TO weight
STORE marstat TO marst
STORE ' TO tunit,tu1,tu2,tu3
SELECT 4
FIND &tserno
IF .NOT. EOF()
STORE schoolname TO tunit
STORE 0 TO tweight
SELECT 3
LOCATE FOR serno = tserno
DELETE

```

ELSE
  SELECT 5
  STORE unit1 TO tu1
  STORE unit2 TO tu2
  STORE unit3 TO tu3
  STORE .F. TO resolved
  STORE 1 TO loopcnt

  * determine which of the requested units
  * will be examined next (1st, 2nd, or 3rd)
  DO WHILE .NOT. resolved .AND. loopcnt < 4
    IF loopcnt = 1
      STORE tu1 TO tu
    ELSE
      IF loopcnt = 2
        STORE tu2 TO tu
      ELSE
        STORE tu3 TO tu
      ENDIF
    ENDIF

    * check if requested unit is available,
    SELECT 3
    LOCATE FOR unitname = tu .AND. serno # tserno,
      .AND. .NOT. DELETED()
    IF .NOT. EOF()
      STORE serno TO idno
      SELECT 2
      LOCATE FOR serno = idno .AND. .NOT. DELETED()

      IF .NOT. EOF()
        * someone else from the requested unit is to be
        * moved check if someone else has requested the
        * same unit
        SELECT 3
        STORE .F. TO done

        DO WHILE .NOT. EOF() .AND. .NOT. done
          IF .NOT. DELETED()
            IF serno # tserno
              IF unit1 = tu .OR. unit2 = tu,
                .OR. unit3 = tu
                STORE .T. TO done
              ENDIF
            ENDIF
          ENDIF
          SKIP
        ENDDO
      ENDIF
    ENDIF
  ENDIF

```



```

IF .NOT. done
* nobody else requests the same unit, requested
* unit is granted, record is marked for deletion
  STORE tu TO tunit

  DO CASE
    CASE loopcnt = 1
      STORE 0 TO tweight
    CASE loopcnt = 2
      STORE 1 TO tweight
    CASE loopcnt = 3
      STORE 2 TO tweight
  ENDCASE

  STORE .T. TO granted
  LOCATE FOR serno = tserno
  DELETE
  SELECT 2
  LOCATE FOR serno = idno
  DELETE
  STORE .T. TO resolved
ELSE
* somebody else requests the same unit, check
* assignment weights to see whom will be given
* the preference to
  DO WHILE ((.NOT. EOF()) .AND. (preference))
    IF .NOT. DELETED()
      IF serno # tserno
        IF unit1 = tu .OR. unit2 = tu,
          .OR. unit3 = tu
          IF asnweight > weight
            STORE .F. TO preference
          ELSE
            IF asnweight = weight
              IF marst = 'S',
                .AND. marstat # 'S'
                STORE .F. TO preference
            ENDIF
          ENDIF
        ENDIF
      ENDIF
    ENDIF
  ENDDO

  IF preference
    * officer requested this unit has the
    * preference, unit is granted, record
    * is marked for deletion
    STORE tu TO tunit

```

```

DO CASE
  CASE loopcnt = 1
    STORE 0 TO tweight
  CASE loopcnt = 2
    STORE 1 TO tweight
  CASE loopcnt = 3
    STORE 2 TO tweight
ENDCASE

STORE .T. TO granted
LOCATE FOR serno = tserno
DELETE
SELECT 2
LOCATE FOR serno = idno
DELETE
STORE .T. TO resolved
ENDIF
ENDIF
ENDIF
STORE loopcnt + 1 TO loopcnt
ENDDO

IF .NOT. resolved
  * unresolved assignment, defer it for the next pass
  STORE '*' TO tunit
  STORE 3 TO tweight
  STORE .F. TO ok
ENDIF
ENDIF

* build record in the file containing the assignments
SELECT 1
APPEND BLANK
REPLACE serno WITH tserno, rank WITH trank,,
       source WITH tsource, specialty WITH tspec
REPLACE unitname WITH tunit, asnweight WITH tweight,,
       asgndate WITH CTOD(tdate)
SELECT 5
SKIP
ENDDO

```

* second pass

IF .NOT. ok

* unresolved assignments exist

SELECT 6

USE unit INDEX unit

@ 8,23 SAY ' SECOND PASS'

STORE .F. TO resolved

STORE .F. TO deletion

SELECT 1

* build temporary file containing unresolved assignments

COPY TO file8 FOR unitname = '*' .AND. rank = '05'

SELECT 5

USE file8

* get one record at a time and try to satisfy request

DO WHILE .NOT. EOF()

STORE ' ' TO tsource

STORE serno TO tserno

STORE specialty TO tspec

STORE source TO tsource

STORE ' ' TO tunit

SELECT 2

* find requested units and store them into memvars

LOCATE FOR serno = tserno

STORE ' ' TO tu

STORE wcfinished TO twc

STORE unit1 TO tu1

STORE unit2 TO tu2

STORE unit3 TO tu3

STORE .F. TO found

STORE .T. TO requested

GO TOP

* find available units

DO WHILE .NOT. EOF() .AND. .NOT. found

IF .NOT. DELETED()

IF serno # tserno

STORE serno TO idno

* unit available

```

DO CASE
  CASE unitname = tu1
    * unit is the 1st requested
    STORE .T. TO found
    STORE unitname TO tunit
    STORE 0 TO tweight
  CASE unitname = tu2
    * unit is the 2nd requested
    STORE .T. TO found
    STORE unitname TO tunit
    STORE 1 TO tweight
  CASE unitname = tu3
    * unit is the 3rd requested
    STORE .T. TO found
    STORE unitname TO tunit
    STORE 2 TO tweight
  OTHERWISE
    * found available unit is not requested
    IF unitname = 'OAB'
      IF twc
        STORE .F. TO requested
      ENDIF
    ELSE
      STORE unitname TO tu
      SELECT 6
      FIND &tu
      IF category = 'S'
        IF twc
          STORE .F. TO requested
        ENDIF
      ELSE
        STORE .F. TO requested
      ENDIF
    ENDIF
  SELECT 2
ENDCASE
ENDIF
ENDIF
SKIP
ENDDO

IF found
  * unit available is one of the requested units, mark
  *   record for deletion, so that it will not be
  *   encountered again
  LOCATE FOR serno = idno
  DELETE
  STORE .T. TO resolved

```

```

ELSE
  IF .NOT. requested
    * available unit is not requested
    * mark record for deletion
    LOCATE FOR serno = idno
    STORE unitname TO tunit
    STORE 3 TO tweight
    STORE .T. TO resolved
    DELETE
  ELSE
    @ 22,18 SAY 'OFFICER CANNOT BE ASSIGNED' GET tserno
  ENDIF
ENDIF
SELECT 1
FIND &tserno
IF resolved
  * update record in assigned file
  REPLACE unitname WITH tunit, asnweight WITH tweight
ELSE
  * no available unit found, delete created in assigned
  * file record. Officer will remain in the same unit
  DELETE
  STORE .T. TO deletion
ENDIF
SELECT 5
STORE .F. TO resolved
SKIP
ENDDO

ENDIF
IF deletion
  SELECT 1
  PACK
ENDIF

* update USERLOG file
SELECT 7
USE userid
LOCATE FOR password = UPPER(psw)
SELECT 8
USE userlog
APPEND BLANK
REPLACE username WITH G->username, task WITH 'ASSIGNMENTS',,
  progame WITH 'ASSIGN05', logdate WITH DATE(),,
  logtime WITH TIME()
CLOSE DATABASES
DELETE FILE file5.dbf
DELETE FILE file6.dbf
DELETE FILE file7.dbf
DELETE FILE file3.dbf
RETURN

```

***** PROGRAM MKAUXFL5 *****

* This program builds the required auxiliary files for the
* processing of the assignments of the lieutenant colonels

DO window

* build file containing personal data and requests

* for the lieutenant colonels under assignment

SELECT 1

USE serves

SELECT 2

USE officer

@ 8,30 SAY 'FILE5

COPY TO file1 FOR rank = '05'

SELECT 1

COPY TO file2 FOR DATE() - enroldate >= 600

SELECT 3

USE file2

SELECT 4

USE file1

JOIN WITH C TO file5 FOR serno = C->serno ,

FIELDS serno,rank,source,specialty,asnweight,marstat,,
unitname

@ 8,45 SAY 'READY'

CLOSE DATABASES

DELETE FILE file1.dbf

DELETE FILE file2.dbf

SELECT 5

USE requests

SELECT 6

USE file5

@ 9,30 SAY 'FILE6

JOIN WITH E TO file6 FOR serno = E->serno,

FIELDS serno,rank,source,specialty,asnweight,marstat,,
unitname,unit1,unit2,unit3,wcfinished

CLOSE DATABASES

DELETE FILE file5.dbf

SELECT 7

USE file6

COPY TO file5

@ 9,45 SAY 'READY'

@ 10,30 SAY 'FILE7

COPY TO file7

@ 10,45 SAY 'READY'

CLOSE DATABASES

RETURN

***** PROGRAM ASSIGN06 *****

* This program performs the assignments of the colonels

* build necessary auxiliary files
DO mkauxfl6

* assign colonels to units and staffs in which they can
* serve by combining requests and assignment weights

DO window1
@ 5,21 SAY ' ASSIGNMENTS FOR THE COLONELS'
@ 6,21 SAY ' ARE BEING PROCESSED'

* open required for the processing files

SELECT 1
USE assigned INDEX assigned
SELECT 2
USE file5
SELECT 3
USE file6
SELECT 4
USE file7

STORE .T. TO ok
@ 8,23 SAY '
@ 8,23 SAY ' FIRST PASS'
STORE '08/15/86' TO tdate

* get each officer under assignment and determine the
* unit to be assigned or defer it for the second pass

DO WHILE .NOT. EOF()
STORE .F. TO granted
STORE .T. TO preference
STORE serno TO tserno
STORE rank TO trank
STORE source TO tsource
STORE specialty TO tspec
STORE asnweight TO weight
STORE marstat TO marst
STORE ' ' TO tunit,tu1,tu2,tu3
STORE unit1 TO tu1
STORE unit2 TO tu2
STORE unit3 TO tu3
STORE .F. TO resolved
STORE 1 TO loopcnt

```

* determine which of the requested units
* will be examined next (1st, 2nd, or 3rd)
DO WHILE .NOT. resolved .AND. loopcnt < 4
  IF loopcnt = 1
    STORE tu1 TO tu
  ELSE
    IF loopcnt = 2
      STORE tu2 TO tu
    ELSE
      STORE tu3 TO tu
    ENDIF
  ENDIF
ENDIF

* check if requested unit is available
SELECT 3
LOCATE FOR unitname = tu .AND. serno # tserno,
      .AND. .NOT. DELETED()
IF .NOT. EOF()
  STORE serno TO idno
  SELECT 2
  LOCATE FOR serno = idno .AND. .NOT. DELETED()

  IF .NOT. EOF()
    * someone else from the requested unit is to be moved
    * check if someone else has requested the same unit
    SELECT 3
    STORE .F. TO done
    DO WHILE .NOT. EOF() .AND. .NOT. done
      IF .NOT. DELETED()
        IF serno # tserno
          IF unit1=tu .OR. unit2=tu .OR. unit3=tu
            STORE .T. TO done
          ENDIF
        ENDIF
      ENDIF
    ENDIF
    SKIP
  ENDDO

  IF .NOT. done
    * nobody else requests the same unit, requested
    * unit is granted, record is marked for deletion
    STORE tu TO tunit

    DO CASE
      CASE loopcnt = 1
        STORE 0 TO tweight
      CASE loopcnt = 2
        STORE 1 TO tweight
      CASE loopcnt = 3
        STORE 2 TO tweight
    ENDCASE
  
```



```

STORE .T. TO granted
LOCATE FOR serno = tserno
DELETE
SELECT 2
LOCATE FOR serno = idno
DELETE
STORE .T. TO resolved

```

ELSE

```

* somebody else requests the same unit, check
* assignment weights to see whom will be given
* the preference to
DO WHILE ((.NOT. EOF()) .AND. (preference))
  IF .NOT. DELETED()
    IF serno # tserno
      IF unit1 = tu .OR. unit2 = tu,
        .OR. unit3 = tu
        IF asnweight > weight
          STORE .F. TO preference
        ELSE
          IF asnweight = weight
            IF marst = 'S',
              .AND. marstat # 'S'
              STORE .F. TO preference
            ENDIF
          ENDIF
        ENDIF
      ENDIF
    ENDIF
  ENDIF
  SKIP
ENDDO

```

IF preference

```

* officer requested this unit has the
* preference, unit is granted, record
* is marked for deletion
STORE tu TO tunit

```

DO CASE

```

CASE loopcnt = 1
  STORE 0 TO tweight
CASE loopcnt = 2
  STORE 1 TO tweight
CASE loopcnt = 3
  STORE 2 TO tweight

```

ENDCASE

```

STORE .T. TO granted
LOCATE FOR serno = tserno
DELETE

```

```

                SELECT 2
                LOCATE FOR serno = idno
                DELETE
                STORE .T. TO resolved
            ENDIF
        ENDIF
    ENDIF
    ENDIF
    STORE loopcnt + 1 TO loopcnt
ENDDO

IF .NOT. resolved
* unresolved assignment, defer it for the next pass
    STORE '*' TO tunit
    STORE 3 TO tweight
    STORE .F. TO ok
ENDIF

* build record in the file containing the assignments
SELECT 1
APPEND BLANK
REPLACE serno WITH tserno, rank WITH trunk,,
        source WITH tsource, specialty WITH tspec
REPLACE unitname WITH tunit, asnweight WITH tweight,,
        asgndate WITH CTOD(tdate)
SELECT 4
SKIP
ENDDO

* second pass

IF .NOT. ok
* unresolved assignments exist
    @ 8,23 SAY '                SECOND PASS'
    STORE .F. TO deletion

    SELECT 1
    * get one record at a time and try
    * to satisfy unsatisfied requests
    GO TOP
    STORE 0 TO cnt

    DO WHILE .NOT. EOF()
        IF unitname = '*'
            STORE serno TO tserno
            STORE ' ' TO tunit,tu1,tu2,tu3
            SELECT 3
            GO TOP
        
```

```

* find requested units and store them into memvars
LOCATE FOR serno = tserno
STORE unit1 TO tu1
STORE unit2 TO tu2
STORE unit3 TO tu3
STORE .F. TO found
* find available units
SELECT 2
GO TOP

DO WHILE ((.NOT. EOF()) .AND. (.NOT. found))
  IF .NOT. DELETED()
    IF serno # tserno
      * unit available
      DO CASE
        CASE unitname = tu1
          * unit is the 1st requested
          STORE unitname TO tunit
          STORE 0 TO tweight
          DELETE
          STORE .T. TO found
        CASE unitname = tu2
          * unit is the 2nd requested
          STORE unitname TO tunit
          STORE 1 TO tweight
          DELETE
          STORE .T. TO found
        CASE unitname = tu3
          * unit is the 3rd requested
          STORE unitname TO tunit
          STORE 2 TO tweight
          DELETE
          STORE .T. TO found
        OTHERWISE
          * found available unit is not requested
          STORE cnt + 1 TO cnt
      ENDCASE
    ENFIF
  ENDIF
  SKIP
ENDDO

IF found
  * available unit is one of the requested
  SELECT 1
  REPLACE unitname WITH tunit,,
  asnweight WITH tweight
ENDIF
ENDIF
SELECT 1
SKIP
ENDDO

```

```

SELECT 1
GO TOP

DO WHILE .NOT. EOF()
    IF unitname = '*'
        STORE serno TO tserno
        STORE ' ' TO tunit
        SELECT 2
        LOCATE FOR serno # tserno .AND. .NOT. DELETED()
        IF .NOT. EOF()
            STORE unitname TO tunit
            DELETE
            SELECT 1
            REPLACE unitname WITH tunit
        ELSE
            SELECT 1
            DELETE
            STORE .T. TO deletion
            @ 24,20 SAY 'OFFICER CANNOT BE ASSIGNED' GET tserno
        ENDIF
    ENDIF
    SELECT 1
    SKIP
ENDDO

ENDIF
IF deletion
    SELECT 1
    PACK
ENDIF

* update USERLOG file
SELECT 7
USE userid
LOCATE FOR password = UPPER(psw)
SELECT 8
USE userlog
APPEND BLANK
REPLACE username WITH G->username, task WITH 'ASSIGNMENTS',,
        progname WITH 'ASSIGN06', logdate WITH DATE(),,
        logtime WITH TIME()

CLOSE DATABASES
DELETE FILE file5.dbf
DELETE FILE file6.dbf
DELETE FILE file7.dbf
RETURN

```

***** PROGRAM MKAUXFL6 *****

* This program builds the required auxiliary files for the
 * processing of the assignments of the colonels

DO window

* build file containing personal data and
 * requests for the colonels under assignment

```

SELECT 1
USE serves
SELECT 2
USE officer
@ 8,30 SAY 'FILE5 .....
```

COPY TO file1 FOR rank = '06'

```

SELECT 1
COPY TO file2 FOR DATE() - enroldate >= 300
```

```

SELECT 3
USE file2
```

```

SELECT 4
USE file1
```

```

JOIN WITH C TO file5 FOR serno = C->serno ,
      FIELDS serno,rank,source,specialty,asnweight,marstat,,
              unitname
```

@ 8,45 SAY 'READY'

CLOSE DATABASES

DELETE FILE file1.dbf

DELETE FILE file2.dbf

SELECT 5

USE requests

SELECT 6

USE file5

@ 9,30 SAY 'FILE6

```

JOIN WITH E TO file6 FOR serno = E->serno,
      FIELDS serno,rank,source,specialty,asnweight,marstat,,
              unitname,unit1,unit2,unit3,wcfinished
```

CLOSE DATABASES

DELETE FILE file5.dbf

SELECT 7

USE file6

COPY TO file5

@ 9,45 SAY 'READY'

@ 10,30 SAY 'FILE7

COPY TO file7

@ 10,45 SAY 'READY'

CLOSE DATABASES

RETURN

E. PROGRAMS PRODUCING THE REQUIRED LISTS AND REPORTS

***** PROGRAM REPORTS *****

* This program controls the report generation operations

```
CLEAR
PUBLIC repcode
DO rgmenu

DO WHILE repcode # 9
    DO CASE
        CASE repcode = 1
            DO list1
        CASE repcode = 2
            DO list2
        CASE repcode = 3
            DO list3
        CASE repcode = 4
            DO list4
        CASE repcode = 5
            DO list5
        CASE repcode = 6
            DO list6
        CASE repcode = 7
            DO report1
        CASE repcode = 8
            DO report2
    ENDCASE

    DO rgmenu
ENDDO

RETURN
```

***** PROGRAM LIST1 *****

* This program provides display or printer output of the
* assignments for a specific rank

CLEAR

DO window2

STORE ' ' TO trank

@ 5,22 SAY 'LIST OF SCHEDULED ASSIGNMENTS FOR THE'

@ 6,23 SAY ' REQUESTED RANK IS GOING TO BE PRINT '

@ 14,23 SAY ' SPECIFY RANK ==>' GET trank,
PICTURE '99'

READ

@ 14,23 SAY ' '

@ 8,23 SAY ' AUXILIARY FILE REQUIRED'

@ 9,23 SAY ' FOR THE PROCESSING IS BEING BUILT '

@ 10,23 SAY ' WAIT

SELECT 1

USE officer INDEX officer

SELECT 2

USE serves INDEX serves

SELECT 3

USE auxfile

SELECT 5

USE userid

SELECT 6

USE userlog

SELECT 4

USE assigned INDEX assigned

DO WHILE .NOT. EOF()

IF rank = trank

STORE ' ' TO tname

STORE ' ' TO tsunit,tdunit

STORE ' ' TO tduty

STORE serno TO tserno

STORE unitname TO tdunit

STORE asgndate TO tdate

SELECT 1

FIND &tserno

STORE name TO tname

SELECT 2

FIND &tserno

STORE unitname TO tsunit

SELECT 3

APPEND BLANK

REPLACE serno WITH tserno,rank WITH trank,,

name WITH tname,unitid WITH tsunit,dunit WITH tdunit,,

date WITH tdate,duty WITH tduty

ENDIF

```

        SELECT 4
        SKIP
    ENDDO

    @ 8,47 SAY 'READY'
    SELECT 3
    STORE ' ' TO answer
    @ 14,23 SAY '          PRINTER OUTPUT? (Y/N) ==>' GET answer
    READ
    IF UPPER(answer) = 'Y'
        @ 14,23 SAY '
        @ 12,23 SAY '          SWITCH ON YOUR PRINTER'
        SET PRINT ON
        @ 13,23 SAY '          HIT ANY KEY  TO CONTINUE'
        SET CONSOLE OFF
        WAIT
        SET CONSOLE ON
    ENDIF
    CLEAR
    REPORT FORM mklist1
    IF UPPER(answer) = 'Y'
        SET PRINT OFF
    ENDIF
    GO TOP
    DELETE NEXT 500
    PACK
    SELECT 5
    LOCATE FOR password = UPPER(psw)
    SELECT 6
    APPEND BLANK
    REPLACE username WITH E->username, task WITH 'LIST # 1',,
        progame WITH 'LIST1', logdate WITH DATE(),,
        logtime WITH TIME()
    CLOSE DATABASES
    RETURN

```


***** PROGRAM LIST2 *****

* This program provides display or printer output of the
* officers serving in a specific unit

```

CLEAR
DO window2
STORE ' ' TO tunit

@ 5,23 SAY ' LIST OF OFFICERS SERVING IN A'
@ 6,23 SAY ' SPECIFIC UNIT IS GOING TO BE PRINT '
@ 14,23 SAY ' SPECIFY UNIT ==>' GET tunit
READ
@ 14,23 SAY '
@ 8,23 SAY ' AUXILIARY FILE REQUIRED'
@ 9,23 SAY ' FOR THE PROCESSING IS BEING BUILT '
@ 10,23 SAY ' WAIT .....

USE officer INDEX officer ALIAS lookup
SELECT 2
USE auxfile
SELECT 5
USE userid
SELECT 6
USE userlog
SELECT 3
USE serves INDEX serves
SET RELATION TO serno INTO lookup

DO WHILE .NOT. EOF()
    IF unitname = tunit
        SELECT 2
        APPEND BLANK
        REPLACE serno WITH C->serno,rank WITH lookup->rank,,
            name WITH lookup->name,unitid WITH C->unitname,,
            date WITH C->enroldate,duty WITH C->duty
    ENDIF

    SELECT 3
    SKIP
ENDDO

@ 8,47 SAY 'READY'
SELECT 2
STORE ' ' TO answer
@ 14,23 SAY ' PRINTER OUTPUT? (Y/N) ==>' GET answer
READ
    
```

```

IF UPPER(answer) = 'Y'
  @ 14,23 SAY '
  @ 12,23 SAY '          SWITCH ON YOUR PRINTER'
  SET PRINT ON
  @ 13,23 SAY '          HIT ANY KEY  TO CONTINUE'
  SET CONSOLE OFF
  WAIT
  SET CONSOLE ON
ENDIF

CLEAR
REPORT FORM mklist2

IF UPPER(answer) = 'Y'
  SET PRINT OFF
ENDIF

GO TOP
DELETE NEXT 500
PACK
SELECT 5
LOCATE FOR password = UPPER(psw)
SELECT 6
APPEND BLANK
REPLACE username WITH E->username, task WITH 'LIST # 2',,
       progname WITH 'LIST2', logdate WITH DATE(),,
       logtime WITH TIME()
CLOSE DATABASES
RETURN

```

***** PROGRAM LIST3 *****

* This program provides display or printer output of the
 * Artillery officers in some requested order

CLEAR
 DO window2
 STORE 0 TO order

@ 5,23 SAY ' LIST OF OFFICERS'
 @ 6,23 SAY ' IN SOME ORDER IS GOING TO BE PRINT'
 @ 8,23 SAY ' POSSIBLE ORDERS'
 @ 9,23 SAY ' SENIORITY (RANK):..... 1'
 @ 10,23 SAY ' ALPHABETICAL:..... 2'
 @ 11,23 SAY ' SPECIALTY:..... 3'
 @ 12,23 SAY ' RANK + ALPHABETICAL:..... 4'
 @ 14,23 SAY ' SPECIFY ORDER ==>' GET order,
 PICTURE '9' RANGE 1,49

READ
 @ 5,23 SAY ' AUXILIARY FILE REQUIRED'
 @ 6,23 SAY ' FOR THE PROCESSING IS BEING BUILT '
 @ 8,23 SAY ' WAIT '
 @ 9,23 SAY ' '
 @ 10,23 SAY ' '
 @ 11,23 SAY ' '
 @ 12,23 SAY ' '
 @ 14,23 SAY ' '

USE serves INDEX serves ALIAS lookup
 SELECT 2
 USE auxfile
 SELECT 4
 USE userid
 SELECT 5
 USE userlog
 SELECT 3
 USE officer INDEX officer,names,ospecial,compound

DO CASE
 CASE order = 1
 SET INDEX TO officer
 CASE order = 2
 SET INDEX TO names
 CASE order = 3
 SET INDEX TO ospecial
 CASE order = 4
 SET INDEX TO compound
 ENDCASE

SET RELATION TO serno INTO lookup

```

DO WHILE .NOT. EOF()
  SELECT 2
  APPEND BLANK
  REPLACE rank WITH C->rank,name WITH C->name,,
         source WITH C->source,specialty WITH C->specialty,,
         unitid WITH lookup->unitname,marstat WITH C->marstat
  SELECT 3
  SKIP
ENDDO

@ 8,47 SAY 'READY'
SELECT 2
STORE ' ' TO answer
@ 14,23 SAY '          PRINTER OUTPUT? (Y/N) ==>' GET answer
READ

IF UPPER(answer) = 'Y'
  @ 14,23 SAY '
  @ 12,23 SAY '          SWITCH ON YOUR PRINTER'
  SET PRINT ON
  @ 13,23 SAY '          HIT ANY KEY  TO CONTINUE'
  SET CONSOLE OFF
  WAIT
  SET CONSOLE ON
ENDIF

CLEAR
REPORT FORM mklist3

IF UPPER(answer) = 'Y'
  SET PRINT OFF
ENDIF

GO TOP
DELETE NEXT 500
PACK

SELECT 4
LOCATE FOR password = UPPER(psw)
SELECT 5
APPEND BLANK
REPLACE username WITH D->username, task WITH 'LIST # 3',,
       progname WITH 'LIST3', logdate WITH DATE(),,
       logtime WITH TIME()
CLOSE DATABASES
RETURN

```

***** PROGRAM LIST4 *****

* This program provides display or printer output of the
 * Artillery officers of some requested rank

CLEAR
 DO window2
 STORE ' ' TO trank

@ 5,23 SAY ' LIST OF OFFICERS OF SOME'
 @ 6,23 SAY 'REQUESTED RANK IS GOING TO BE PRINT'
 @ 14,23 SAY ' SPECIFY RANK ==>' GET trank PICTURE 'XX'
 READ
 @ 5,23 SAY ' AUXILIARY FILE REQUIRED '
 @ 6,23 SAY ' FOR THE PROCESSING IS BEING BUILT '
 @ 8,23 SAY ' WAIT

USE serves INDEX serves ALIAS lookup
 SELECT 2
 USE auxfile
 SELECT 4
 USE userid
 SELECT 5
 USE userlog
 SELECT 3
 USE officer INDEX officer
 SET RELATION TO serno INTO lookup

DO WHILE .NOT. EOF()
 IF rank = trank
 SELECT 2
 APPEND BLANK
 REPLACE rank WITH C->rank,name WITH C->name,,
 source WITH C->source,specialty WITH C->specialty,,
 unitid WITH lookup->unitname,marstat WITH C->marstat
 ENDIF

SELECT 3
 SKIP
 ENDDO

@ 8,47 SAY 'READY'
 SELECT 2
 STORE ' ' TO answer
 @ 14,23 SAY ' PRINTER OUTPUT? (Y/N) ==>' GET answer
 READ

```

IF UPPER(answer) = 'Y'
  @ 14,23 SAY '
  @ 12,23 SAY '          SWITCH ON YOUR PRINTER'
  SET PRINT ON
  @ 13,23 SAY '          HIT ANY KEY  TO CONTINUE'
  SET CONSOLE OFF
  WAIT
  SET CONSOLE ON
ENDIF

CLEAR
REPORT FORM mklist4

IF UPPER(answer) = 'Y'
  SET PRINT OFF
ENDIF

GO TOP
DELETE NEXT 500
PACK

SELECT 4
LOCATE FOR rassword = UPPER(psw)
SELECT 5
APPEND BLANK
REPLACE username WITH D->username, task WITH 'LIST # 4',,
        progame WITH 'LIST4', logdate WITH DATE(),,
        logtime WITH TIME()
CLOSE DATABASES
RETURN

```

***** PROGRAM LIST5 *****

* This program provides display or printer output of
* the Artillery battalion commanders

CLEAR
DO window2

@ 5,23 SAY ' LIST OF THE BATTALION COMMANDERS '
@ 6,23 SAY ' IS GOING TO BE PRINT '
@ 8,23 SAY ' AUXILIARY FILE REQUIRED '
@ 9,23 SAY ' FOR THE PROCESSING IS BEING BUILT '
@ 10,23 SAY ' WAIT '

USE serves INDEX serves ALIAS lookup
SELECT 2
USE auxfile
SELECT 4
USE useid
SELECT 5
USE userlog
SELECT 3
USE officer INDEX officer
SET RELATION TO serno INTO lookup

DO WHILE .NOT. EOF()
IF rank = '05'
IF lookup->duty = 'CDR'
SELECT 2
APPEND BLANK
REPLACE rank WITH C->rank,name WITH C->name,,
source WITH C->source,,
specialty WITH C->specialty,,
marstat WITH C->marstat,serno WITH C->serno
REPLACE unitid WITH lookup->unitname,,
date WITH lookup->enroldate
ENDIF
ENDIF

SELECT 3
SKIP
ENDDO

@ 10,47 SAY 'READY'
SELECT 2
STORE ' ' TO answer
@ 14,23 SAY ' PRINTER OUTPUT? (Y/N) ==>' GET answer
READ

```

IF UPPER(answer) = 'Y'
  @ 14,23 SAY '
  @ 12,23 SAY '          SWITCH ON YOUR PRINTER'
  SET PRINT ON
  @ 13,23 SAY '          HIT ANY KEY  TO CONTINUE'
  SET CONSOLE OFF
  WAIT
  SET CONSOLE ON
ENDIF

CLEAR
REPORT FORM mklist5

IF UPPER(answer) = 'Y'
  SET PRINT OFF
ENDIF

GO TOP
DELETE NEXT 500
PACK

SELECT 4
LOCATE FOR password = UPPER(psw)
SELECT 5
APPEND BLANK
REPLACE username WITH D->username, task 'LIST # 5',,
        progame WITH 'LIST5', logdate WITH DATE(),,
        logtime WITH TIME()

CLOSE DATABASES
RETURN

```


***** PROGRAM LIST6 *****

* This program provides display or printer output for the
 * officers serving outside the Artillery Branch

CLEAR
 DO window2

@ 5,23 SAY ' LIST OF OFFICERS SERVING OUTSIDE '
 @ 6,23 SAY ' THE BRANCH IS GOING TO BE PRINT '
 @ 8,23 SAY ' AUXILIARY FILE REQUIRED '
 @ 9,23 SAY ' FOR THE PROCESSING IS BEING BUILT '
 @ 10,23 SAY ' WAIT '

USE serves INDEX serves ALIAS lookup
 SELECT 2
 USE auxfile
 SELECT 4
 USE userid
 SELECT 5
 USE userlog
 SELECT 3
 USE officer INDEX officer
 SET RELATION TO serno INTO lookup

DO WHILE .NOT. EOF()
 IF lookup->unitname = 'OAB'
 SELECT 2
 APPEND BLANK
 REPLACE rank WITH C->rank,name WITH C->name,,
 source WITH C->source,specialty WITH C->specialty,,
 marstat WITH C->marstat,serno WITH C->serno
 REPLACE unitid WITH lookup->unitname,,
 date WITH lookup->enroldate
 ENDIF
 SELECT 3
 SKIP
 ENDDO

@ 10,47 SAY 'READY'
 SELECT 2
 STORE ' ' TO answer
 @ 14,23 SAY ' PRINTER OUTPUT? (Y/N) ==>' GET answer
 READ

```

IF UPPER(answer) = 'Y'
  @ 14,23 SAY '
  @ 12,23 SAY '          SWITCH ON YOUR PRINTER'
  SET PRINT ON
  @ 13,23 SAY '          HIT ANY KEY  TO CONTINUE'
  SET CONSOLE OFF
  WAIT
  SET CONSOLE ON
ENDIF

CLEAR
REPORT FORM mklist6

IF UPPER(answer) = 'Y'
  SET PRINT OFF
ENDIF

GO TOP
DELETE NEXT 500
PACK

SELECT 4
LOCATE FOR password = UPPER(psw)
SELECT 5
APPEND BLANK
REPLACE username WITH D->username, task WITH 'LIST # 6',,
        progame WITH 'LIST6', logdate WITH DATE(),,
        logtime WITH TIME()

CLOSE DATABASES
RETURN

```

***** PROGRAM REPORT1 *****

* This program provides display or printer output containing
 * all service data concerning a specific officer

CLEAR
 DO window2
 STORE ' ' TO tserno.

@ 5,22 SAY 'SERVICE TIME REPORT FOR THE REQUESTED'
 @ 6,23 SAY ' OFFICER IS GOING TO BE PRINT '
 @ 14,23 SAY ' ENTER SERIAL # ==>' GET tserno
 READ
 @ 14,23 SAY ' '

SELECT 3
 USE userid
 SELECT 4
 USE userlog
 SELECT 1
 USE historic INDEX historic
 SELECT 2
 USE officer INDEX officer
 FIND &tserno

IF .NOT. EOF()
 STORE name TO tname
 STORE source TO tsource
 STORE specialty TO tspec
 STORE DATE() TO tdate
 SELECT 1
 STORE ' ' TO answer
 @ 14,23 SAY ' PRINTER OUTPUT? (Y/N) ==>' GET answer
 READ

IF UPPER(answer) = 'Y'
 @ 14,23 SAY '
 @ 12,23 SAY ' SWITCH ON YOUR PRINTER'
 @ 13,23 SAY ' HIT ANY KEY TO CONTINUE'
 SET CONSOLE OFF
 WAIT
 SET CONSOLE ON
 SET PRINT ON
ENDIF
CLEAR

```

? '          ABD/HAGS                      DATE: ',tdate
? '          -----                      -----
? '
? '          OFFICER'S SERVICE TIME REPORT'
? '          =====
? '
? '          SERIAL # : ',tserno,'NAME : ',tname
? '          SOURCE : ',tsource,'          SPECIALTY : ',tspec
? '
? '  DATE    TRANSACTION RANK    UNIT          ORDER ID'
? '-----
LIST transdate,transtype,rank,unit,orderid,
    FOR serno = tserno

IF UPPER(answer) = 'Y'
    SET PRINT OFF
    SELECT 3
    LOCATE FOR password = UPPER(psw)
    SELECT 4
    APPEND BLANK
    REPLACE username WITH C->username,,
           task WITH 'REPORT # 1',,
           progname WITH 'REPORT1', logdate WITH DATE(),,
           logtime WITH TIME()
ENDIF

ELSE
    @ 14,23 SAY 'OFFICER DOES NOT EXIST IN DATABASE'
    DO delay
ENDIF

CLOSE DATABASES
RETURN

```

***** PROGRAM REPORT2 *****

* This program provides display or printer output
 * concerning the status of a specific officer

```

CLEAR
DO window2
STORE ' ' TO tserno

@ 5,23 SAY ' STATUS REPORT FOR THE REQUESTED'
@ 6,23 SAY ' OFFICER IS GOING TO BE PRINT '
@ 14,23 SAY ' ENTER SERIAL # ==>' GET tserno
READ

@ 14,23 SAY '
STORE DATE() TO tdate

USE serves INDEX serves ALIAS lookup
SELECT 2
USE officer INDEX officer
SET RELATION TO serno INTO lookup
FIND &tserno

IF .NOT. EOF()
  STORE ' ' TO answer
  @ 14,23 SAY ' PRINTER OUTPUT? (Y/N) ==>' GET answer
  READ

  IF UPPER(answer) = 'Y'
    @ 14,23 SAY '
    @ 12,23 SAY ' SWITCH ON YOUR PRINTER'
    SET PRINT ON
    @ 13,23 SAY ' HIT ANY KEY TO CONTINUE'
    SET CONSOLE OFF
    WAIT
    SET CONSOLE ON
  ENDIF

  CLEAR
  
```

```

? ' ABD/HAGS DATE: ', tdate
? ' -----
? '
? ' OFFICER'S STATUS REPORT'
? ' =====
? '
? ' NAME : ', name
? ' SERIAL NUMBER : ', serno
? ' RANK : ', rank
? ' NOMINATION YEAR: ', nomyear
? ' SOURCE : ', source
? ' SPECIALTY : ', specialty
? ' MARITAL STATUS : ', marstat
? ' CHILDREN : ', children
? ' WORKING WIFE : ', workwife
? ' ORIGIN : ', origcity, ', ', origcounty
? ' UNIT : ', lookup->unitname
? ' ENROLMENT DATE : ', lookup->enroldate
? ' DUTY : ', lookup->duty
? '
IF UPPER(answer) = 'Y'
    SET PRINT OFF
ENDIF

SELECT 3
USE userid
LOCATE FOR password = UPPER(psw)
SELECT 4
USE userlog
APPEND BLANK
REPLACE username WITH C->username,,
        task WITH 'REPORT # 2',,
        progame WITH 'REPORT2', logdate WITH DATE(),,
        logtime WITH TIME()

ELSE
    @ 14,23 SAY 'OFFICER DOES NOT EXIST IN DATABASE'
    DO delay
ENDIF

CLOSE DATABASES
RETURN

```

F. MISCELLANEOUS PROGRAMS

***** PROGRAM GRFLAG *****

* This program displays on the screen the Hellenic Flag

```
SET COLOR TO W+
CLEAR
STORE '      ' TO blank1
STORE '                                ' TO blank2
STORE '                                ' TO blank3
STORE 7 TO r
STORE 28 TO c
@ 7,18 GET blank1
@ 7,24 GET blank1
@ 8,18 GET blank1
@ 8,24 GET blank1
@ 10,18 GET blank1
@ 10,24 GET blank1
@ 11,18 GET blank1
@ 11,24 GET blank1

DO WHILE r < 13
    @ r,c GET blank3
    STORE r+2 TO r
ENDDO

STORE 18 TO c

DO WHILE r < 17
    @ r,c GET blank2
    STORE r+2 TO r
ENDDO

SET COLOR TO W
RETURN
```

***** PROGRAM DELAY *****

* This program provides a small delay necessary for displaying
* various program messages on the screen

```
STORE 0 TO k

DO WHILE k < 40
    STORE k + 1 TO k
ENDDO

RETURN
```

***** PROGRAM DBSTITLE *****

* This program displays on the screen the title of our
* database system

CLEAR

```
@ 8,18 SAY 'IMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM, '
@ 9,18 SAY ':      HABS/ARTILLERY BRANCH DIRECTORATE      : '
@ 10,18 SAY ':      MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM : '
@ 11,18 SAY ':      :      :      :      :      :      :      :      :      : '
@ 12,18 SAY ':      :      :      :      :      :      :      :      :      : '
@ 13,18 SAY ':      :      :      :      :      :      :      :      :      : '
@ 14,18 SAY ':      :      :      :      :      :      :      :      :      : '
@ 15,18 SAY ':      :      :      :      :      :      :      :      :      : '
@ 16,18 SAY ':      :      :      :      :      :      :      :      :      : '
@ 17,18 SAY 'HMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM< '
```

SET COLOR TO W+

```
@ 11,28 SAY 'PERSONNEL DATABASE SYSTEM'
@ 12,22 SAY 'FOR PROCESSING THE ANNUAL ASSIGNMENTS'
@ 13,32 SAY 'OF THE OFFICERS'
```

SET COLOR TO W

SET CONSOLE OFF

WAIT

SET CONSOLE ON

RETURN

***** PROGRAM FRAME *****

* This program displays a small frame on the screen

```
@ 8,43 SAY 'IMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM,'
@ 9,43 SAY ': NEW RECORD IN HISTORIC FILE : '
@ 10,43 SAY ': MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM : '
@ 11,43 SAY ': : '
@ 12,43 SAY ': : '
@ 13,43 SAY ': : '
@ 14,43 SAY ': : '
@ 15,43 SAY ': : '
@ 16,43 SAY ': : '
@ 17,43 SAY 'HMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM<'
RETURN
```

***** PROGRAM WINDOW *****

* displays a frame on the screen into which messages for the
* user are placed informing him about the processing

```
CLEAR
@ 4,20 SAY 'IMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM,'
@ 5,20 SAY ': TEMPORARY FILES FOR THE : '
@ 6,20 SAY ': ASSIGNMENT PROCESSING ARE BEING BUILT : '
@ 7,20 SAY ': MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM : '
@ 8,20 SAY ': : '
@ 9,20 SAY ': : '
@ 10,20 SAY ': : '
@ 11,20 SAY ': : '
@ 12,20 SAY ': : '
@ 13,20 SAY ': : '
@ 14,20 SAY ': : '
@ 15,20 SAY 'HMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM<'
RETURN
```

***** PROGRAM WINDOW1 *****

* This program displays a window on the screen

```
CLEAR
@ 4,18 SAY 'IMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM,'
@ 5,18 SAY ': : '
@ 6,18 SAY ': : '
@ 7,18 SAY ': : '
@ 8,18 SAY ': : '
@ 9,18 SAY 'HMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM<'
RETURN
```


G. SAMPLE LISTS AND REPORTS (DATABASE SYSTEM OUTPUT)

Page No. 1
04/28/86

ABD/HAGS

LIST OF SCHEDULED ASSIGNMENTS FOR THE REQUESTED RANK

SERIAL #	RANK	NAME	FROM UNIT	TO UNIT	DUE DATE
22860	02	Zimbo Carl P	GMB	212 FAB	08/15/86
22873	02	Zets Frank B	AS	12 HAB	08/15/86
22888	02	White Henry V	12 HAB	213 MHAB	08/15/86
23029	02	Topping Scott A	GMB	122 FAB	08/15/86
23050	02	Sugai Steve D	122 FAB	GMB	08/15/86
23112	02	Plott Charles B	GMB	211 FAB	08/15/86
23141	02	Murphy Tom S	23 OB	112 FAB	08/15/86
23229	02	Kondler Ronald F	AS	GMB	08/15/86
23285	02	Otondo Jeff M	GMB	23 OB	08/15/86
23326	02	Weber Alex T	211 FAB	AS	08/15/86
23359	02	Yeates Jim N	213 MHAB	AS	08/15/86
23392	02	Timar Gordon C	112 FAB	113 MHAB	08/15/86
23405	02	Taylor Leroy A	113 MHAB	GMB	08/15/86
23595	02	Amason Frank L	212 FAB	GMB	08/15/86
23849	02	Powers Albert G	111 FAB	OAB	08/15/86
23864	02	Kilgore Michael F	OAB	111 FAB	08/15/86
23890	02	Ratt Bruno K	OAB	OAB	08/15/86
24007	02	Jeronomo Mike W	121 FAB	22 HAB	08/15/86
24023	02	Johnson Mark J	22 HAB	212 FAB	08/15/86
24046	02	Davidson Jack B	212 FAB	122 FAB	08/15/86
24061	02	Fallis John M	122 FAB	121 FAB	08/15/86
24156	02	Linhard Harry C	211 FAB	21 FA/AB	08/15/86
24175	02	Voris John D	21 FA/AB	211 FAB	08/15/86

AD-A174 137

IMPLEMENTATION OF A PERSONNEL DATABASE SYSTEM
PERFORMING THE ANNUAL REASS (U) NAVAL POSTGRADUATE
SCHOOL MONTEREY CA I G STROUZAS 28 JUN 86

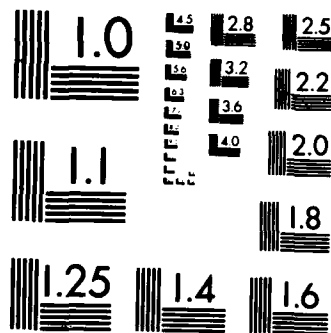
3/3

UNCLASSIFIED

F/G 5/9

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

ABD/HAGS

LIST OF OFFICERS SERVING IN THE REQUESTED UNIT

RANK	SERIAL#	NAME	UNIT	DUTY	ENROL. DATE
07	20060	Morris Robert G	ARTC	CDR	05/05/85
06	20178	Adams Edward B	ARTC	DEPUTY CDR	05/22/85
05	20273	Anzini Daniel D	ARTC	FAASS	07/26/84
05	20300	Lipori Gerald A	ARTC	TAB/CDR	07/18/84
05	20391	McRae James R	ARTC	TAB/CDR	07/12/84
05	20531	Nicely Marvin L	ARTC	TAB/CDR	07/20/85
04	20785	Russo James D	ARTC	G2	08/18/85
04	20927	Qualls Terry D	ARTC	G3	07/29/84
04	20948	Miller Jacj T	ARTC	G1	07/28/85
04	21138	Krupski Joseph K	ARTC	HSB/CDR	07/20/83
04	21249	Rogers Alex B	ARTC	TCAO	08/25/85
03	21289	Mallon Patrick F	ARTC	B/CDR	08/27/84
03	21423	Trend Ted M	ARTC	B/CDR	08/30/84
03	21492	Aston John S	ARTC	B/CDR	08/29/84
03	21562	Salvo John H	ARTC	B/CDR	08/31/84
03	21628	Ryan Peter G	ARTC	B/CDR	08/26/82
03	21689	Papas Chris J	ARTC	B/CDR	08/27/84
03	21733	Strouzas John G	ARTC	B/CDR	08/30/84
03	21903	Solomos James D	ARTC	B/CDR	08/29/84
03	21956	Sercia John M	ARTC	B/CDR	08/31/85
01	24402	Lucas Mike T	ARTC	P/CDR	08/01/85
01	24405	Williby Richard F	ARTC	P/CDR	08/01/85
01	24423	Grace Bob K	ARTC	P/CDR	08/01/85
01	24426	Stetson Jeff C	ARTC	P/CDR	08/01/85
01	24436	Nagal David A	ARTC	P/CDR	08/01/85
01	24444	Dunlop Tom S	ARTC	P/CDR	08/01/85
01	24458	Kohn Daniel P	ARTC	P/CDR	08/01/85
01	24459	Eakin William F	ARTC	P/CDR	08/01/85
01	24472	Clanin John V	ARTC	P/CDR	08/01/85
01	24473	Henderson Lester N	ARTC	P/CDR	08/01/85
01	24491	Lubin Patrick H	ARTC	P/CDR	08/01/85
01	24495	Naef George J	ARTC	P/CDR	08/01/85
01	24786	Monsen Harry K	ARTC	MO	08/25/84
01	24832	Mixter Jason K	ARTC	MO	08/30/85
01	24847	Townsend Jeff P	ARTC	MO	08/28/85
01	24851	Lucky Thomas S	ARTC	PAO	08/29/83
01	24929	Knubis James P	ARTC	PAO	08/29/84
01	25016	Cline William R	ARTC	PAO	08/30/85

ABD/HAGS

LIST OF ARTILLERY OFFICERS IN THE REQUESTED ORDER

RANK	NAME	SOURCE	SPECIALTY	UNIT-NAME	MARITAL ST.
02	Abigil Frank P	MA	C	GMB	S
01	Abston Mike R	MA	C	212 FAB	S
01	Acevedo William D	MA	C	113 MHAB	S
06	Adams Edward B	MA	C	ARTC	M
03	Adcok Jerry T	MA	C	21 FA/AB	M
04	Alders Edward R	MA	C	OAB	M
02	Allen Douglas W	NCOS	A	223 MHAB	M
03	Allen Kirk R	MA	C	221 FAB	S
01	Alonso Jack N	NCOS	T	23 OB	M
02	Amason Frank L	MA	C	212 FAB	M
03	Amman George D	MA	C	AS	M
01	Anasini George S	MA	C	112 FAB	S
02	Anderson Moore J	MA	C	221 FAB	S
02	Aney John T	MA	C	13 OB	M
03	Annelo Philip R	MA	C	122 FAB	M
07	Anton John T	MA	C	OAB	M
06	Anway James R	MA	C	OAB	M
05	Anzini Daniel D	MA	C	ARTC	M
04	Appel John G	MA	C	WC	M
03	Arima James J	MA	C	GMB	M
05	Ariss Bruce F	MA	C	AS	S
04	Armout Paul G	MA	C	22 HAB	M
04	Armstrong David K	MA	C	WC	M
04	Arnal Robert J	MA	C	OAB	M
05	Arnold Michael C	MA	C	212 FAB	M
03	Arundel Mike A	NCOS	T	113 MHAB	D
03	Aston John S	MA	C	ARTC	M
03	Atkins Robert H	MA	C	121 FAB	M
04	Auburn James P	MA	C	111 FAB	M
01	Austin Dick B	MA	C	AS	S
05	Avery Adam S	MA	C	221 FAB	M
01	Axelrod Jack Z	NCOS	C	111 FAB	S
03	Babbitt Almon P	MA	C	211 FAB	M
03	Baer Dan F	NCOS	C	OAB	M
05	Balum Douglas N	MA	C	21 FA/AB	M
03	Beam Alan K	MA	C	112 FAB	M
01	Beaty Jackson L	MA	C	122 FAB	S
05	Bell Richard K	MA	C	OAB	M
04	Benison Michael J	MA	C	OAB	M
07	Billeb James W	MA	C	OAB	M
04	Biondi Daniel M	MA	C	223 MHAB	M

Page No. 1
04/28/86

ABD/HAGS

LIST OF ARTILLERY OFFICERS OF THE REQUESTED RANK

RANK	NAME	SOURCE	SPECIALTY	MARITAL	ST.	UNIT-NAME
05	Knight Allen S	MA	C	M		OAB
05	Klose Edwin A	MA	C	M		OAB
05	Nakata Isaac E	MA	C	M		GMB
05	Franklin Adams P	MA	C	M		ABD/HAGS
05	Cabral David T	MA	C	M		OAB
05	Bell Richard K	MA	C	M		OAB
05	Anzini Daniel D	MA	C	M		ARTC
05	Ariss Bruce F	MA	C	S		AS
05	Gray Joseph W	MA	C	M		OAB
05	Jarecki Edward L	MA	C	M		ABD/HAGS
05	Harvey Steeve B	MA	C	N		OAB
05	Carl Michael S	MA	C	N		22 HAB
05	Lipori Gerald A	MA	C	D		ARTC
05	Naylor Stephen G	MA	C	M		121 FAB
05	Wurtz Thomas D	MA	C	M		OAB
05	Yadon Robert J	MA	C	M		ABD/HAGS
05	Zeller Donald L	MA	C	M		OAB
05	Yee Kalvin B	MA	C	S		OAB
05	Plott Jeff C	MA	C	M		222 FAB
05	Tranel James A	MA	C	M		AS
05	Trotter Richard E	MA	C	M		122 FAB
05	Valentino John P	MA	C	S		OAB
05	Urie Larry H	MA	C	M		AS
05	Thoreson Alex G	MA	C	M		OAB
05	Smith Steven G	MA	C	M		113 MHAB
05	Johnes Robert P	MA	C	M		111 FAB
05	Poleti Felix A	MA	C	M		12 HAB
05	Nash Thomas S	MA	C	W		OAB
05	McRae James R	MA	C	M		ARTC
05	Goral George B	MA	C	M		211 FAB
05	Drew Alan M	MA	C	S		OAB
05	Cosio Frank L	MA	C	M		213 MHAB
05	Avery Adam S	MA	C	M		221 FAB
05	Balum Douglas N	MA	C	M		21 FA/AB
05	Jeffrey Harold D	MA	C	M		OAB
05	Nicasio Michael F	MA	C	M		OAB
05	Wagner Chris F	MA	C	D		ABD/HAGS
05	Lippman Bill J	MA	C	M		OAB
05	Jagoe Donald H	MA	C	S		223 MHAB
05	Grace Bruce P	MA	C	M		11 FA/AB
05	Draper George R	MA	C	M		OAB

Page No. 1
04/28/86

ABD/HAGS

LIST OF THE ARTILLERY BATTALION COMMANDERS

SERIAL #	RANK	NAME	UNIT	ENROL. DATE
20255	05	Nakata Isaac E	GMB	07/19/84
20294	05	Carl Michael S	22 HAB	07/16/85
20305	05	Naylor Stephen G	121 FAB	07/11/84
20332	05	Plott Jeff C	222 FAB	07/24/84
20343	05	Trotter Richard E	122 FAB	07/13/84
20365	05	Smith Steven G	113 MHAB	07/22/84
20376	05	Johnes Robert P	111 FAB	07/21/84
20379	05	Poleti Felix A	12 HAB	07/23/84
20392	05	Goral George B	211 FAB	07/22/84
20403	05	Cosio Frank L	213 MHAB	07/18/84
20408	05	Avery Adam S	221 FAB	07/17/84
20412	05	Balum Duglas N	21 FA/AB	07/23/84
20449	05	Jagoe Donald H	223 MHAB	07/18/85
20455	05	Grace Bruce P	11 FA/AB	07/11/85
20476	05	Johnson Mark L	123 MHAB	07/18/85
20495	05	Arnold Michael C	212 FAB	07/21/85
20500	05	Knapp Thomas L	112 FAB	07/19/85

ABD/HAGS

LIST OF ARTILLERY OFFICERS SERVING OUTSIDE THE BRANCH

RANK	SERIAL #	NAME	UNIT	ENROL. DATE
07	20017	Calaunan Tend G	OAB	05/20/85
07	20019	Billeb James W	OAB	05/15/85
07	20020	Anton John T	OAB	05/18/85
07	20031	Magnelli Harold P	OAB	05/25/85
07	20067	Prevat Randy K	OAB	05/12/85
06	20081	Horton Alex B	OAB	06/01/85
06	20088	Gordan Michael A	OAB	06/10/85
06	20097	Nickel George W	OAB	06/20/85
06	20109	Wapper Alfred D	OAB	06/18/85
06	20121	Anway James R	OAB	06/12/85
06	20151	McNett John R	OAB	06/21/85
06	20173	Colvin Edgar A	OAB	06/15/85
06	20195	Wright Richard T	OAB	06/17/84
06	20196	Yancy Roy J	OAB	06/12/84
06	20208	Luk William P	OAB	06/17/84
06	20217	Lacy David A	OAB	06/18/84
06	20226	Plantz Joe M	OAB	06/19/84
05	20240	Knight Allen S	OAB	07/12/84
05	20248	Klose Edwin A	OAB	07/15/85
05	20269	Cabral David T	OAB	07/27/84
05	20270	Bell Richard K	OAB	07/13/84
05	20277	Gray Joseph W	OAB	07/18/85
05	20290	Harvey Steeve B	OAB	07/10/84
05	20313	Wurtz Thomas D	OAB	07/12/85
05	20324	Zeller Donald L	OAB	07/10/85
05	20329	Yee Calvin B	OAB	07/20/85
05	20351	Valentino John P	OAB	07/19/85
05	20360	Thoreson Alex G	OAB	07/09/85
05	20382	Nash Thomas S	OAB	07/16/84
05	20399	Drew Alan M	OAB	07/13/84
05	20421	Jeffrey Harold D	OAB	07/17/85
05	20427	Nicasio Michael F	OAB	07/20/84
05	20442	Lippman Bill J	OAB	07/08/85
05	20468	Draper George R	OAB	07/10/85
05	20483	Cook William K	OAB	07/06/85
05	20511	Kasper James K	OAB	07/10/85
05	20519	Kennedy John B	OAB	07/05/85
04	20556	Lucky Thomas N	OAB	08/20/84
04	20570	Benison Michael J	OAB	08/23/84
04	20582	Alders Edward R	OAB	08/18/84
04	20596	Arnal Robert J	OAB	08/21/84

Page No. 2
04/28/86

ABD/HAGS

LIST OF ARTILLERY OFFICERS SERVING OUTSIDE THE BRANCH
=====

RANK	SERIAL #	NAME	UNIT	ENROL. DATE
----	-----	-----	-----	-----
04	20607	Durbin James C	OAB	08/16/84
04	20619	Kaseman Timothy A	OAB	08/12/84
04	20641	Holste Robert W	OAB	08/24/85
04	20655	Nagano Patrick K	OAB	08/28/84
04	20666	Combie Edward P	OAB	08/29/85
04	20670	Saleh William R	OAB	08/22/84
04	20685	Wyler Bill A	OAB	04/26/85
04	20697	Trapl Mark W	OAB	08/26/85
04	20715	Scilk Alan B	OAB	08/27/85
04	20722	Scott Calvin D	OAB	08/26/85
04	20723	Hoss Jack C	OAB	08/28/84
04	20728	Goodrich John A	OAB	08/27/84
04	20734	Tally Chris S	OAB	08/27/84
04	20736	Concon Stephen J	OAB	08/17/85
04	20749	Emerson Burt F	OAB	08/24/84
04	20756	Jefferson Jack L	OAB	08/25/84

ABD/HAGS

DATE: 04/28/86

OFFICER'S SERVICE TIME REPORT
=====

SERIAL # : 20204 NAME : Wechsler Thomas J
SOURCE : MA SPECIALTY : C

DATE	TRANSACTION	RANK	UNIT	ORDER ID
07/25/64	NOMINATION	01	MA	F.440/11/HAGS
08/02/64	ASSIGNMENT	01	AS	F.400/23/ABD/HAGS
08/01/65	ASSIGNMENT	01	ARTC	F.400/46/ABD/HAGS
08/12/66	ASSIGNMENT	01	111 FAB	F.400/32/ABD/HAGS
07/26/67	PROMOTION	02	111 FAB	F.440/41/ABD/HAGS
08/22/70	ASSIGNMENT	02	223 MHAB	F.400/54/ABD/HAGS
07/28/72	PROMOTION	03	223 MHAB	F.440/33/HAGS
08/25/74	ASSIGNMENT	03	12 MHAB	F.400/44/ABD/HAGS
08/01/77	PROMOTION	04	12 HAB	F.440/22/HAGS
08/28/77	ASSIGNMENT	04	13 OB	F.400/41/ABD/HAGS
08/01/79	ASSIGNMENT	04	WC	F.333/20/ABD/HAGS
07/25/80	ASSIGNMENT	04	AC/11 ID	F.400/21/ABD/HAGS
05/18/81	PROMOTION	05	AC/11 ID	F.440/46/HAGS
07/23/82	ASSIGNMENT	05	21 FA/AB	F.400/24/ABD/HAGS
06/22/85	ASSIGNMENT	05	ABD/HAGS	F.400/13/ABD/HAGS
04/20/86	PROMOTION	06	ABD/HAGS	F.400/18/ABD/HAGS

ABD/HAGS

DATE: 04/28/86

OFFICER'S STATUS REPORT
=====

NAME : Trotter Richard E
SERIAL NUMBER : 20343
RANK : 05
NOMINATION YEAR: 66
SOURCE : MA
SPECIALTY : C
MARITAL STATUS : M
CHILDREN : 2
WORKING WIFE : .F.
ORIGIN : city19 , county10
UNIT : 122 FAB
ENROLMENT DATE : 07/13/84
DUTY : CDR

LIST OF REFERENCES

1. David Kroenke, Database Systems, 2d ed., SRS Inc., 1983.
2. Richard Fairley, Software Engineering Concepts, McGraw-Hill Book Co., 1985.
3. Jeffrey D. Ullman, Database Systems, 2d ed., Computer Science Press Inc., 1982.
4. Allan Simpson, Understanding dBASE III, SYBEX Inc., 1985.
5. dBASE III User Manual, Ashton-Tate, 1984.
6. James Senn, Analysis and Design of Information Systems, MacGraw-Hill Book Co., 1984

INITIAL DISTRIBUTION LIST

	No. Copies
1. Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2. Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5000	2
3. Computer Technology Curricular Office Code 37 Naval Postgraduate School Monterey, California 93943-5100	1
4. CDR L. Rawlinson, Code 52Rv Department of Computer Science Naval Postgraduate School Monterey, California 93943-5100	1
5. CDR G. S. Baker, Code 52B Department of Computer Science Naval Postgraduate School Monterey, California 93943-5100	2
6. Hellenic Army General Staff/G3 Stratopedo Papagou Holargos Athens Greece	2
7. Hellenic Army General Staff Artillery Branch Directorate Stratopedo Papagou Holargos Athens Greece	2
8. Defense and Army Attache Embassy of Greece 2228 Mass. Avenue, N.W. Washington DC, 20008	1
9. MJ Strouzas Ioannis Hellenic Army General Staff Stratopedo Papagou Holargos Athens Greece	6
10. LTC Tsagaris Panagiotis Hellenic Army General Staff Stratopedo Papagou Holargos Athens Greece	1

- | | |
|--|---|
| 11. CDR Solomos Demosthenes
Naval Postgraduate School
SMC 2157 | 1 |
| 12. CP Karatasios Lambros
Naval Postgraduate School
SMC 1831 | 1 |
| 13. LT Anastasatos Costas
Hellenic Navy General Staff
Stratopedo Papagou
Holargos Athens Greece | 1 |
| 14. CP Vassiliou Antonios
Naval Postgraduate School
SMC 1817 | 1 |

END

12-86

DTIC